DOI:
10.1007/s41060-017-0082-x

Document Version
Peer reviewed version

Noname manuscript No.
(will be inserted by the editor)

Preventing the diffusion of information to vulnerable users while preserving PageRank*

Grigorios Loukides · Robert Gwadera

Received: date / Accepted: date

Abstract Limiting the diffusion of information in social networks is important in viral marketing and computer security. To achieve this, existing works aim to prevent the diffusion of information to as many nodes as possible, by deleting a given number of edges. Thus, they adopt a collective approach and quantify the impact of deletion on the graph, based on the number of deleted edges.

In this work, we propose a selective approach which quantifies the impact of edge deletion based on PageRank. Our approach allows specifying the nodes to which information diffusion should be prevented and their maximum allowable activation probability. Furthermore, it performs edge deletion while avoiding drastic changes to the ability of the network to propagate information. To realize our approach, we propose a measure that captures changes, caused by deletion, to the PageRank distribution of the graph. Our measure is called *PageRank-Harm* (PRH) and quantifies the contribution of an incoming edge (u_l, u) to the PageRank score of the node u. Based on PRH, we define the following optimization problem: Given a subset of nodes and a threshold, find a subset of edges that has minimum PRH and whose deletion limits the activation probability of each specified node to at most the threshold.

We show that the problem can be modeled as a *Submodular Set Cover* (*SSC*) problem and design an approximation algorithm, based on the well-known approximation algorithm for *SSC*. Furthermore, we develop an iterative heuristic that has similar effectiveness but also enables significant computational savings. Moreover, we propose a lazy edge selection technique that is used to improve the efficiency of both our approximation algorithm and the iterative heuristic, without affecting their effectiveness. Experiments on real and synthetic data show the effectiveness and efficiency of our methods.

Keywords Information diffusion, Influence, Linear threshold model, PageRank, Submodular Set Cover

1 Introduction

Social networks have become a ubiquitous communication infrastructure, which enables the diffusion (propagation) of information to a very large number of users. For instance, social networks are used by businesses who promote products through viral marketing campaigns [11,17,24] but also by malicious users who spread computer malware [25, 26, 40,43]. Therefore, controlling the diffusion of information is becoming an important task in multiple domains, including viral marketing and computer security. In the most common setting, the diffusion starts from a small subset of users who aim to *activate* their friends. The activated friends of these users attempt to activate their own friends, and the diffusion process proceeds similarly until no new users are activated. The diffusing information comes in different forms, such as a link to the website of a new product or to a malicious website to download malware. Typically, the social network is represented as a graph, the initial users correspond to a subset of nodes called *seeds*, and the activation probabilities of nodes are computed according to a diffusion model [24].

Recently, many works [25, 26, 38, 40] focused on limiting the diffusion of potentially harmful information, by mo-

* This paper is an extended version of the paper "*Limiting the diffusion of information by a selective PageRank-preserving approach*" that was presented at the IEEE International Conference on Data Science and Advanced Analytics (DSAA) 2016.

G. Loukides
Department of Informatics, King's College London
Tel: +44 (0)20 7848 8496
E-mail: gloukides@acm.org

R. Gwadera
School of Computer Science & Informatics, Cardiff University
E-mail: GwaderaR@cardiff.ac.uk

difying the graph before the start of the diffusion process. These works aim to find a subset of k edges, whose deletion by a decision maker (*operator*) reduces the expected number of activated nodes at the end of the process (*spread*) as much as possible. However, they consider a rather limited setting, since they: (I) adopt a *collective* approach (i.e., assume that the diffusing information can harm all users), and (II) use the number of deleted edges to quantify the impact of edge deletion on the graph.

In this work, we consider the problem of limiting information diffusion through edge deletion, in a new setting. Specifically, we propose a *selective* approach that allows specifying the nodes to which information should not be diffused (*vulnerable nodes*) and their maximum allowable activation probability. This flexibility is important in marketing when there are certain classes of users, based on demographics, location, or health condition, that may be harmed by the diffusing information about a product [19], or form and spread negative opinions about it [9]. In addition, our approach quantifies the impact of edge deletion on the graph using *PageRank* [4, 6], a fundamental model of information propagation based on network topology [3, 42]. This allows performing deletion while preserving the ability of the network to propagate information. For example, when applied to the graph of Fig. 1(a), our approach favors the deletion of the edge e_3 instead of e_1. The deletion of e_3 allows the propagation of information to more nodes (e.g., from the vulnerable nodes u_1, u_2 to u_8) and causes a smaller change to the PageRank distribution of the graph, as can be seen in Fig. 1(b).

Our approach reduces the activation probability $\mathcal{P}_v$ of each vulnerable node v to at most a threshold $max\mathcal{P}$, while aiming to preserve the PageRank distribution of the graph. The activation probabilities are computed by the Linear Threshold (LT) [24] model, a well-established model of the diffusion of potentially harmful information [25, 26]. The LT model captures the "threshold behavior" of users, in which a user takes the action that has been taken by a sufficiently large fraction of their friends. The threshold $max\mathcal{P}$ is a simple, application-dependent measure of significance (alike the *minimum support* threshold in pattern mining), which models the maximum allowable activation probability for each vulnerable node. The selection of $max\mathcal{P}$ and of vulnerable nodes is performed based on domain knowledge (e.g., customer vulnerability analysis and policies [39]).

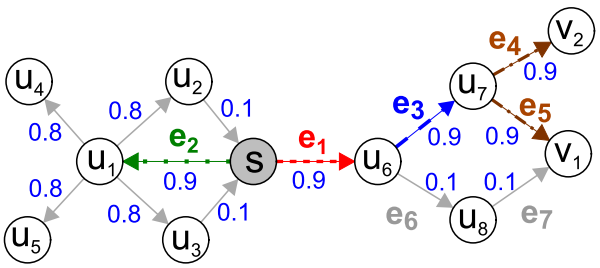
Enforcing our approach is challenging, because: (I) There is an exponential number of edge subsets that can be deleted. (II) There are dependencies between edges, which affect the activation probability of nodes. Specifically, the deletion of an edge (u_l, u) reduces the activation probability of all non-seed nodes reachable from u, because these nodes can no longer be activated by a path that contains (u_l, u). (III) Existing measures (e.g., L_1 distance and KL-divergence) [4] that quantify changes to the PageRank distribution cannot be used as optimization criteria in efficient approximation algorithms. In addition, our approach cannot be enforced by existing methods [25,26] that limit the diffusion of information under the LT model. This is because these methods may not limit the activation probabilities of vulnerable nodes, or they may substantially affect the information propagation on the network. To illustrate this point, we provide Example 1, where we apply the method of [25] with different k values. This method aims to minimize the spread of the diffusing information (expected number of activated nodes), under the LT model, by deleting an edge subset of given size k.

Example 1 Consider the graph of Fig. 1(a), where the seed is s, and the vulnerable nodes are v_1 and v_2. The activation probabilities $\mathcal{P}_{v_1}$ and $\mathcal{P}_{v_2}$ in the LT model are equal to 0.738 and 0.729, respectively, and they need to be limited to at most 0.01. Applying the method of [25] with $k = 1$, deletes $e_2 = (s, u_1)$. This minimizes the spread. However, $\mathcal{P}_{v_1}$ and $\mathcal{P}_{v_2}$ do not change, since all simple paths from s to v_1 and to v_2 are preserved [17]. Thus, in this case, the method does not limit the activation probabilities of vulnerable nodes to the desired threshold. Using $k = 2$, results in deleting $\{e_1, e_2\}$. This reduces $\mathcal{P}_{v_1}$ and $\mathcal{P}_{v_2}$, to zero. However, no information can be propagated from u_1, u_2, or u_3 to the nodes on the right of s. Thus, in this case, the edge deletion method of [25] substantially affects the information propagation on the network.

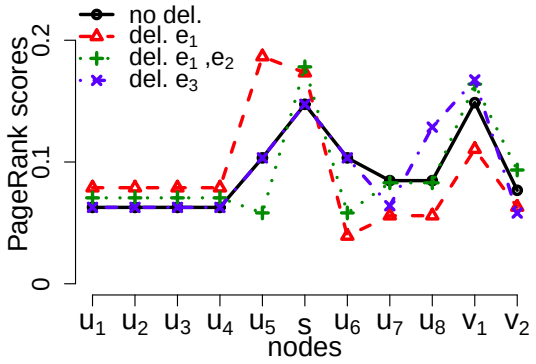
1.1 Contributions

The contributions of our work are summarized as follows.

First, we propose a measure that captures changes, caused by edge deletion, to the PageRank distribution. Our measure, called *PageRank-Harm* (PRH), quantifies the contribution of an incoming edge (u_l, u) to the PageRank score of the node u. Thus, it penalizes the deletion of the edge based on the ratio between the PageRank score and out-degree of the start node (see Eq. 1). For example, $e_1 = (s, u_6)$ has a larger PRH than $e_3 = (u_6, u_7)$ in Fig. 1(a), because s has a larger PageRank score than u_6 (see Fig. 1(b)) and s and u_6 have the same out-degree. Since the PageRank score of each node is distributed equally into its out-neighbors, deleting an edge with large PRH incurs a substantial change to the PageRank scores of many other nodes. For instance, deleting e_1 instead of e_3 causes a larger change to the PageRank scores of the nodes in Fig. 1(a), as shown in Fig. 1(b). Since PageRank is computed recursively (see Eq. 1), these changes propagate in the graph and may also affect the PageRank score of u_l, due to paths from u to u_l. However, as we show, the impact of such changes is small in practice, because the change to the PageRank score of a node diminishes exponentially with the length of the path from u_l. Therefore,



(a)



(b)

edge	PRH	$g_{\mathcal{P}_{v_1}}$	$g_{\mathcal{P}_{v_2}}$	$g_{\mathcal{P}_{v_1,v_2}}$
e_1	0.062	0.738	0.729	1.447
e_2	0.062	0	0	0
e_3	0.043	0.729	0.729	1.447
e_4	0.036	0	0.729	0.719
e_5	0.036	0.729	0	0.728
e_6	0.043	0.009	0	0.009
e_7	0.072	0.009	0	0.009

(c)

Algorithm	Deleted edges	PRH
AGDE	e_3	0.043
IGDE	$\{e_4, e_5\}$	0.071
[25] $k = 2$	$\{e_1, e_2\}$	0.124

(d)

Fig. 1: (a) Graph and edge probabilities; s is a *seed*, and v_1, v_2 are *vulnerable* nodes. The method of [25] with $k = 1$ (resp., $k = 2$) deletes e_2, (resp., $\{e_1, e_2\}$). (b) The PageRank distribution of the graph in Fig. 1(a) before and after deleting e_1, $\{e_1, e_2\}$, and e_3. (c) PRH, *path probability gain* $g_{\mathcal{P}_{v_1}}$ and $g_{\mathcal{P}_{v_2}}$ used in IGDE, and *aggregate path probability gain* $g_{\mathcal{P}_{v_1,v_2}}$ used in AGDE. (d) The deleted edges and PRH for AGDE, IGDE, and the method of [25] with $k = 2$, when applied to Example 1.

PRH can be used as a heuristic to preserve the PageRank distribution of the graph. In addition, we show that the PRH measure can be incorporated into efficient approximation algorithms.

Second, we formally define the optimization problem of *PageRank-preserving Edge Deletion* (*PED*). The problem requires finding an edge subset whose deletion: (I) minimizes changes to the PageRank distribution of the graph according to PRH, and (II) limits the activation probability of each vulnerable node to at most $max\mathcal{P}$. We also prove that *PED* is NP-hard.

Third, we show that *PED*, for a single vulnerable node, can be modeled as a *Submodular Set Cover* (*SSC*) [14, 41] problem. This allows developing an approximation algorithm based on the well-known logarithmic approximation algorithm for *SSC* [41]. Our algorithm, called Greedy Delete Edges (GDE), finds an edge subset iteratively. In each iteration, it selects the edge with the minimum ratio between PRH and *path probability gain*, which quantifies the benefit of selecting the edge in terms of decreasing the activation probability $\mathcal{P}_v$ of the vulnerable node. The objective is to select an edge that decreases $\mathcal{P}_v$ without substantially affecting PageRank. When the deletion of the selected edges can limit $\mathcal{P}_v$ to at most $max\mathcal{P}$, these edges are deleted and the algorithm stops. GDE finds an edge subset whose PRH is larger than that of the optimal solution by at most a logarithmic factor, which depends on the PRH and the path probability gain of the subset.

Fourth, we propose two algorithms for *PED*, when there are multiple vulnerable nodes. The first is an approximation algorithm, called Aggregate Greedy Delete Edges (AGDE). AGDE differs from GDE in the following aspects: (I) It uses *aggregate path probability*. This function allows determining when the activation probability of each vulnerable node does not exceed $max\mathcal{P}$, and it is used in the stopping criterion of AGDE. (II) It uses the gain in aggregate path probability after selecting an edge (*aggregate path probability gain*) in its edge selection criterion. Aggregate path probability gain quantifies the benefit of selecting the edge in terms of decreasing the activation probability of all vulnerable nodes whose activation probability exceeds $max\mathcal{P}$ simultaneously. AGDE achieves a logarithmic approximation ratio, which depends on the PRH and the aggregate path probability gain of the selected edges. Our experiments show that AGDE finds near-optimal solutions (see Fig. 5a). The second algorithm, Iterative Greedy Delete Edges (IGDE), iterates over the vulnerable nodes, in decreasing order of their activation probability, and applies GDE to approximate

the *PED* problem for one vulnerable node per iteration. As a heuristic to minimize the number of deleted edges, IGDE deals with nodes with large activation probability first. Such nodes "cover" many other vulnerable nodes, because they typically require deleting many edges and the deletion of an edge (u_l, u) reduces the activation probability of all vulnerable nodes reachable from u. IGDE is up to two orders of magnitude faster than AGDE, because the deleted edges in an iteration are not considered again, and it produces solutions of similar quality, as shown in our experiments. To illustrate AGDE and IGDE, we provide Example 2.

Example 2 AGDE and IGDE were applied to Example 1, using $max\mathcal{P} = 0.01$. AGDE selected the edge e_3 in Fig. 1(a), which has the minimum ratio between PRH and aggregate path probability gain $g_{\mathcal{P}_{v_1,v_2}}$ (see Fig. 1(c)). The deletion of e_3 limits $\mathcal{P}_{v_1}$ and $\mathcal{P}_{v_2}$ to at most 0.01, thus AGDE deleted e_3. IGDE considered v_1 first, since $\mathcal{P}_{v_1}$ is larger than $\mathcal{P}_{v_2}$, and selected e_5. This is because e_5 has the minimum ratio between PRH and path probability gain $g_{\mathcal{P}_{v_1}}$ among the edges $\{e_1, e_3, e_5, e_6, e_7\}$, whose deletion decreases $\mathcal{P}_{v_1}$ (see Fig. 1(c)). The deletion of e_5 limits $\mathcal{P}_{v_1}$ to at most 0.01, thus IGDE deleted e_5. Then, IGDE considered v_2 and deleted e_4. The deletion of $\{e_4, e_5\}$ limits both $\mathcal{P}_{v_1}$ and $\mathcal{P}_{v_2}$ to at most 0.01. As shown in Fig. 1(d), the solutions of IGDE and the method of *[25]* with $k = 2$ have 65% and 186% larger PRH than that of the solution of AGDE, respectively.

Fifth, we propose a lazy edge selection technique that is used to improve the efficiency of all our algorithms. The technique is inspired by the *Accelerated Greedy* (also referred to as Lazy Greedy) algorithm [34], and it is based on the submodularity of the functions that are used in the edge selection criteria of our algorithms. The lazy edge selection technique enables our algorithms to find the edge with the minimum ratio, while substantially reducing the number of edges that are considered for being selected. Thus, lazy edge selection does not affect the quality of the solutions of the algorithms, while it substantially improves their runtime. In our experiments, we report at least 37% and up to 814% lower runtime for the AGDE algorithm and at least 15% and up to 87% lower runtime for the IGDE algorithm.

1.2 Paper organization

The rest of the paper is organized as follows. Section 2 provides the background. Section 3 introduces the PRH measure. Section 4 presents the *PED* problem. Sections 5 and 6 present our algorithms. Section 7 our lazy edge selection technique. Section 8 presents the experimental evaluation. Section 9 discusses related work. Section 10 concludes the paper.

2 Background

This section presents necessary concepts that are used in our approach, including PageRank [6], the Linear Threshold model [24], and the Accelerated Greedy algorithm [34].

2.1 Preliminaries

Let $G(V, E)$ be a directed graph. V is a set of nodes of size $|V|$, and E is a set of edges of size $|E|$. The set of in-neighbors of a node u is denoted with $n^-(u)$ and has size $|n^-(u)|$, which is referred to as the *in-degree* of u. The set of out-neighbors of u is denoted with $n^+(u)$ and has size $|n^+(u)|$, which is referred to as the *out-degree* of u.

A path $q = [(u_1, u_2), \ldots, (u_{m-1}, u)]$ is an ordered set of edges, which has *length* $|q| = m-1$. A path q in which each node, $u_1, \ldots, u$, is unique (i.e., a path with no cycle) is a *simple* path. A path that starts and ends at the same node is a *cycle* path. We assume simple paths, unless stated otherwise.

Let R and R' be probability distributions represented with the probability vectors $(r_1, \ldots, r_m)$ and $(r'_1, \ldots, r'_m)$, respectively. The distance between the probability distributions R and R' can be quantified using the KL-divergence or the L_1 distance. The L_1 distance quantifies the absolute error between the distributions R and R' as $L_1(R, R') = \sum_{i\in[1,m]} |r_i - r'_i|$, and it is typically used to measure distance between PageRank distributions [4]. The L_1 distance also forms the basis of the following measures: (I) the $Gower$ distance, which is defined as $Gower(R, R') = \frac{1}{m} \cdot L_1(R, R')$, and (II) the *Average Relative Error* (ARE), which is defined as $ARE(R, R') = \frac{1}{m} \cdot \sum_{i\in[1,m]} \frac{|r_i - r'_i|}{r_i}$.

Let U be a universe of elements and 2^U its power set. A set function $f : 2^U \rightarrow \mathbb{R}$ is *non-decreasing*, if $f(X) \leq f(Y)$ for all subsets $X \subseteq Y \subseteq U$, *monotone*, if $f(X) \leq f(X \cup u)$ for each $u \notin X$, and *submodular*, if it satisfies the *diminishing returns* property $f(X \cup \{u\}) - f(X) \geq f(Y \cup \{u\}) - f(Y)$, for all $X \subseteq Y \subseteq U$ and any $u \in U \setminus Y$ [27].

2.2 PageRank

PageRank [6] is a well-established model of information propagation based on network topology [3, 42]. The *PageRank score* of a node u of a graph G is:

$$PR(u, G) = \frac{\alpha}{|V|} + (1 - \alpha) \cdot \sum_{u_l \in n^-(u)} \frac{PR(u_l, G)}{|n^+(u_l)|} \quad (1)$$

where $\alpha \in (0, 1)$ is the *restart probability*, which is usually set to 0.15 [4]. Eq. 1 assumes that each node has out-degree at least 1 (i.e., there are no *dangling* nodes). If there are dangling nodes, we treat them as in [29]. That is, an artificial edge is added between each dangling node and each node of G, including the dangling node itself. For simplicity of

presentation, we henceforth assume that G does not contain dangling nodes. We will write $PR(u)$ for $PR(u, G)$, when G is clear from the context. The *PageRank distribution* of the graph G is denoted with $PR(G)$, and it is defined as the vector of the PageRank scores of all nodes of G [4].

2.3 The Linear Threshold (LT) model

We now review the Linear Threshold (LT) model following [24]. The *edge probability* of an edge (u_l, u) is denoted with $p((u_l, u))$ and reflects how likely a node u that has not been activated before is activated by its active in-neighbor u_l. For each node u, it holds that $\sum_{u_l \in n^-(u)} p((u_l, u)) \leq 1$ [1]. The *path probability* of a path $q = [(u_1, u_2), \ldots, (u_{m-1}, u)]$ is defined as $P(q) = \prod_{e \in q} p(e)$ and reflects how likely u is activated by u_1 through q.

Let $S \subseteq V$ be the set of seeds. Let also $Q_{s,u}$ be the set of paths from a seed s to a non-seed node u of G that do not pass through another seed, and $Q_{S,u} = \cup_{s \in S} Q_{s,u}$. The *activation probability* of u by $Q_{S,u}$ is computed as $\mathcal{P}(u, Q_{S,u}) = \sum_{q \in Q_{S,u}} P(q)$, where $P(q)$ is the path probability of a path q in $Q_{S,u}$ [17]. We denote $\mathcal{P}(u, Q_{S,u})$ with $\mathcal{P}_u$, when $Q_{S,u}$ is clear from the context. We also define the *activation graph* $\tilde{G}_u$ of u as the subgraph of G which is induced by the edges of all paths in $Q_{S,u}$.

The exact computation of $\mathcal{P}_u$ is a $\#P$-hard problem for general graphs [11]. However, the path probability often decreases exponentially with the path length, because the edge probabilities of most paths are uniformly bounded away from 1 [11]. Thus, $\mathcal{P}_u$ can be estimated accurately and efficiently, based on the subset of paths in $Q_{S,u}$ whose seeds are "close" to u [17]. To find these paths, we adapt the depth-first-search-based algorithm of [17]. For each seed, the algorithm iteratively extends each path from the seed and prunes it, if its path probability is lower than a threshold h. Then, $\mathcal{P}_u$ is computed based on the paths from seeds to u that are found by the algorithm, and $\tilde{G}_u$ is constructed as the graph induced by the edges of these paths. The threshold $h \in [0, 1]$ represents the maximum tolerable estimation error and is operator-specified [17]. The impact of the threshold h on our approach is studied in Section 8.

2.4 Accelerated greedy

The Accelerated Greedy algorithm [34] is an approximation algorithm for finding a subset of r elements whose value in a submodular function is maximum. Formally, given a universe of elements U, a monotone non-decreasing submodular function $f : 2^U \rightarrow \mathbb{R}$, and an integer r, the Accelerated Greedy algorithm [34] finds a subset $U' \subseteq U$ such that: (I) $f(U') \geq (1 - \frac{1}{\mathsf{e}}) \cdot f(U'_{OPT})$, and (II) $|U'| = r$, where $U'_{OPT} \subseteq U$ is the subset of U with the maximum value in f and size r, and e is the base of the natural logarithm [34]. In each iteration, the algorithm selects the element of U with the maximum marginal gain (i.e., the element u that causes the largest increase $f(U' \cup \{u\}) - f(U')$, after being added into U'), which is found efficiently using a priority queue. The priority queue is initialized with the marginal gain of each element and is sorted in decreasing order. In the first iteration, the top entry is removed from the queue and its corresponding element, u_1, is added into the subset U'. In the second iteration, the top entry of the queue (i.e., the second topmost entry in the previous iteration) is updated to reflect the addition of u_1 into U'. If the entry stays on the top of the queue (i.e., it still has the largest marginal gain), it is removed from the queue and its corresponding element, u_2, is added into U'. This is because, after the update, the marginal gain of u_2 is at least equal to that of any node in $U \setminus U'$, and, due to submodularity, the marginal gain of any node in $U \setminus U'$ cannot increase. Otherwise, the algorithm updates the current top entry and repeats the process. After adding an element into U', Accelerated Greedy proceeds into the next iteration, which is similar to the second one. In practice, the number of queue updates performed is small, which makes the algorithm efficient, as reported in [10, 30, 31].

3 The PRH measure

The deletion of an edge affects the PageRank score of the end node of the deleted edge, according to Eq. 1. In addition, the PageRank score of this node is distributed into its out-neighbors. Thus, the PageRank scores of these nodes change, and the change is propagated similarly. Therefore, edge deletion may incur a substantial change to the PageRank distribution. Minimizing the change in our problem is challenging, because there are $O(2^{|E|})$ edge subsets that need to be considered, to select the one that causes the minimum change to the PageRank distribution and reduces the activation probability of each vulnerable node to at most $max\mathcal{P}$. However, the number of edge subsets that are considered can be reduced to $O(|E|^2)$, by observing that the problem is similar to the *Submodular Set Cover* (*SSC*) problem [41] (a formal reduction will be presented in Section 5). *SSC* can be efficiently solved using a *greedy approach* [41]. In the case of our problem, the subset of edges $E' \subseteq E$ to be deleted is constructed iteratively by the greedy approach. That is, the edge e that minimizes the ratio of: (I) the distance between $PR(G'_1)$ and $PR(G'_2)$, where G'_1 (respectively, G'_2) is produced from the graph G by deleting E' (respectively, $E' \cup e$), and (II) *aggregate path probability gain* is iteratively added into E' . The objective is to select edges whose deletion incurs a small change to the PageRank

[1] The restriction provided in [24] allows u to remain inactive, after each of its active in-neighbors has attempted to activate u.

distribution and a large reduction in the activation probability of vulnerable nodes. By favoring edges that incur a large reduction in activation probability, it also tends to select a small number of edges.

The greedy approach can employ different measures to quantify the distance between $PR(G'_1)$ and $PR(G'_2)$, such as the L_1 distance, the $Gower$ distance, ARE, and KL-divergence. However, it does not offer approximation guarantees when it employs either of these measures. This is because the greedy approach offers approximation guarantees, only when it employs a *monotone* measure[2] [41], whereas the measures L_1 distance, $Gower$ distance, ARE, and KL-divergence are not monotone, as shown in Example 3.

Example 3 The subgraphs G'_1 and G'_2 of the graph G in Fig. 1(a) are produced by deleting $E' = \{e_1\}$ and $E' \cup e_2 = \{e_1, e_2\}$, respectively. The distance between $PR(G)$ and $PR(G'_1)$ is higher than that between $PR(G)$ and $PR(G'_2)$, according to each of the measures in Fig. 2. Since $e_2 \notin E'$ and the value of each of the measures in Fig. 2 for E' is larger than the value for $E' \cup e_2$, these measures are not monotone.

Subgraph	Deleted edges	L_1	$Gower$	ARE	KL-divergence
G'_1	$E' = \{e_1\}$	0.347	0.032	0.341	0.126
G'_2	$E' \cup e_2 = \{e_1, e_2\}$	0.188	0.017	0.178	0.051

Fig. 2: Existing measures favor the deletion of the edges $E' \cup e_2 = \{e_1, e_2\}$ instead of $E' = \{e_1\}$ from G in Fig. 1(a).

Therefore, we propose PRH, a monotone measure that can be used by the greedy approach to produce approximately optimal solutions. In Section 3.1, we define the PRH of an edge $e = (u, u')$. In Section 3.2, we show that the the deletion of e has a small impact on the PageRank scores of nodes that are far away from u.

In Section 3.3, we define the PRH of a subset of edges, based on the observation that the dependencies among the PRH of these edges are weak. That is, the deletion of an edge $e = (u, u')$ does not substantially affect the PRH of another edge $e' = (\tilde{u}_1, \tilde{u}_2)$ in the subset. Specifically, we show that the change to the PageRank score of $\tilde{u}_1$ decreases exponentially with the length of the path from u to $\tilde{u}_1$.

3.1 The PRH of a single edge

The PRH of an edge $e = (u, u')$ is defined as $PRH(e) = (1-\alpha) \cdot \frac{PR(u,G)}{|n^+(u)|}$, where $PR(u, G)$ is the PageRank score of u in G, $|n^+(u)|$ is the out-degree of u, and α is the restart probability of Eq. 1. Since α is a constant [6], the term $(1-\alpha)$ in $PRH(e)$ is the same for every edge e and can be omitted. We retain this term to highlight the fact that $PRH(e)$ is exactly the contribution of the incoming edge $e = (u, u')$ to the PageRank score of the node u' (i.e., the PageRank "mass" that the edge e passes to u'), according to Eq. 1.

Intuitively, when the edge $e = (u, u')$ has a large PRH, it "passes" a large PageRank mass from u to u'. This is because a large $PRH(e)$ implies that the PageRank score $PR(u, G)$ of u is large and/or its out-degree $|n^+(u)|$ is small. Thus, deleting e changes the PageRank scores of the out-neighbors of u. In addition, the change propagates in the graph and may incur a large change (increase or decrease) to the PageRank scores of other nodes, due to paths to these nodes that start from u [2, 13].

3.2 Impact of deleting a single edge on PageRank

In the following, we examine the impact of deleting an edge $e = (u, u')$ on the PageRank score of a node u^* in detail. Our objective is to show that the effect of edge deletion has a small impact on nodes that are far away from u. Thus, PRH can capture the change to the PageRank scores of most nodes effectively, which makes it a good heuristic to capture changes to the PageRank distribution of the graph.

Let $\delta(u^*) = PR(u^*, G) - PR(u^*, G')$ be the change to the PageRank score of u^*, when the deletion of the edge $e = (u, u')$ from G produces G'. We show that δ^* heavily depends on $PRH(e)$, which implies that deleting edges with small $PRH(e)$ can be used as a heuristic to preserve the PageRank distribution of the graph. Specifically, there are two cases when the edge e is deleted, which are illustrated in Fig. 3:

I u^* is an *out-neighbor* of u, and
  (a) $u^* = u'$, or (b) $u^* \neq u'$.
II u^* is *not* an out-neighbor of u.

We now consider these cases in detail.

Case I Consider the case I(a). Before the deletion of e, the contribution of e to $PR(u^*)$ was $(1-\alpha) \cdot \frac{PR(u,G)}{|n^+(u)|} = PRH(e)$, according to Eq. 1. However, after deleting e, u is no longer an in-neighbor of u^*. Thus, the contribution of e to $PR(u^*)$ is zero. Now consider the case I(b). The deletion of e reduces the out-degree of the node u by one. Thus, the contribution of (u, u^*) to $PR(u^*)$ changes from $(1-\alpha) \cdot \frac{PR(u,G)}{|n^+(u)|}$ to $(1-\alpha) \cdot \frac{PR(u,G')}{|n^+(u)|-1}$. However, in either case, u^* may have a set of in-neighbors other than u, which is denoted with U_L (see Figs. 3I(a) and 3I(b)).

Therefore, $\delta(u^*)$ is computed as in Eq. 2:

$$\delta(u^*) = \begin{cases} PRH(e) + (1-\alpha) \cdot \sum\limits_{u_l \in U_L} \frac{\delta(u_l)}{|n^+(u_l)|} & , u^* = u' \\ PRH(e) - (1-\alpha) \cdot \frac{PR(u,G')}{|n^+(u)|-1} + (1-\alpha) \cdot \sum\limits_{u_l \in U_L} \frac{\delta(u_l)}{|n^+(u_l)|} & , u^* \neq u' \end{cases} \quad (2)$$

[2] The value of a monotone measure for the deletion of E' cannot be larger than that for the deletion of $E' \cup e$.

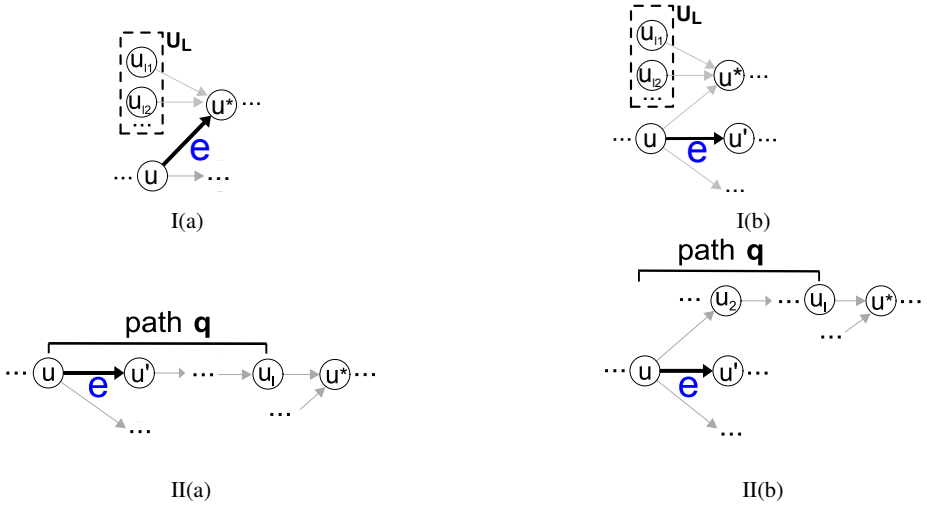


Fig. 3: Cases that summarize the relation between $\delta(u^*)$ and $PRH(e)$. The deleted edge e is in bold, U_L is the set of in-neighbors of u^*, and nodes and edges that are not shown are denoted with "...".

where $\delta(u_l)$ is the change to the PageRank score of a node u_l in U_L, and α is the restart probability of Eq. 2. The proof of Eq. 2 follows easily from Eq. 1 and the definition of PRH, and it is omitted.

Case II The deletion of e changes the PageRank scores of the out-neighbors of u, according to Case I (see Figs. 3II(a) and II(b)), and the change is propagated to other nodes similarly. In particular, $\delta(u^*)$ is computed using Eq. 3:

$$\delta(u^*) = (1-\alpha) \cdot \sum_{u_l \in n^-(u^*)} \frac{\delta(u_l)}{|n^+(u_l)|} \quad (3)$$

which follows from Eq. 2, when u is not an in-neighbor of u^*. Eq. 3 is computed backwards recursively to the out-neighbors of u.

Thus, in Cases I and II, $\delta(u^*)$ is determined by $PRH(e)$ and/or by the change to $PR(u^*)$, caused by the incoming edges to u^*. Furthermore, the change incurred by an edge (u_l, u^*) decreases exponentially with the length of the path from u to u_l. Specifically, given a simple path

$$q = [(u, u'), (u', u'_2) \ldots, (u'_{|q|-1}, u_l)]$$

(see Fig. 3II(a)), we obtain Eq. 4:

$$\delta(u_l) = (1-\alpha)^{|q|-1} \cdot \frac{\delta(u')}{|n^+(u')| \cdot \prod_{r=2}^{|q|-1} |n^+(u'_r)|} \quad (4)$$

by recursively applying Eq. 3 for $\delta(u_l)$ over $u'_{|q|-1}, \ldots, u'_2$. The case of a path q containing a cycle is similar (omitted). Therefore, $\delta(u_l)$ diminishes as we move away from u, and $\delta(u^*)$ heavily depends on $PRH(e)$ in most cases. Consequently, PRH is an effective heuristic to capture changes, caused by edge deletion, to the PageRank scores of nodes.

3.3 The PRH of a subset of edges

We define the PRH of an edge subset $E' \subseteq E$ as

$$PRH(E') = \sum_{e \in E'} PRH(e)$$

where e is an edge in E' that starts from a node u of G and $PRH(e) = (1-\alpha) \cdot \frac{PR(u,G)}{|n^+(u)|}$. Clearly, PRH is monotone since $PRH(E') \leq PRH(E' \cup e')$, for each edge $e' \notin E'$.

The PRH of each edge in E' is computed based on the graph G. This strategy allows our approach to select an edge efficiently, without computing the PageRank distribution of the graph that is produced by the deletion of the currently selected edges. Furthermore, the strategy is effective, because the deletion of a currently selected edge $e = (u, u')$ does not substantially affect the PRH of another edge $e' = (\tilde{u}_1, \tilde{u}_2)$.

This is because $\delta(\tilde{u}_1)$ decreases exponentially with the length of the path from u to $\tilde{u}_1$, since $\delta(\tilde{u}_1)$ is computed by applying Eq. 4 for $u_l = \tilde{u}_1$. Thus, $\delta(\tilde{u}_1)$ is a small fraction of $\delta(u^*)$, which is already small, since $\delta(u^*)$ depends on $PRH(e)$ and our approach selects edges with small PRH. In Section 8, we show that our PRH computation strategy is much more efficient and equally effective as the alternative strategy, which computes $PRH(e)$ on the graph that is produced from G by deleting the currently selected edges.

4 Problem definition

The *PED* problem is defined as follows.

Problem 1 (PageRank-preserving Edge Deletion (*PED*)) Given a graph $G(V, E)$, a threshold $max\mathcal{P}$ in $[0, 1]$, a set of seed nodes S and a set of vulnerable nodes D, such that $S, D \subseteq V$ and $S \cap D = \varnothing$, and the PRH of each edge

$e \in E$, find an edge subset $E' \subseteq E$, so that the following two conditions hold:

(I) $PRH(E')$ is minimum, and
(II) the activation probability $\mathcal{P}_v \leq max\mathcal{P}$, for each node $v \in D$, after the deletion of E' from G.

The problem requires finding an edge subset E' with minimum PRH, whose deletion limits the activation probability $\mathcal{P}_v$ of each vulnerable node v to at most $max\mathcal{P}$. We assume that the activation probability $\mathcal{P}_v$, before edge deletion, is larger than $max\mathcal{P}$, for each vulnerable node v in D. If this condition is not satisfied, v is excluded from D. We also assume that the operator selects the seeds (e.g., using existing methods [17, 24]), as well as the threshold $max\mathcal{P}$ and the vulnerable nodes, based on domain knowledge (e.g., customer vulnerability analysis and policies [39]). In addition, the operator computes the PRH of each edge. The *PED* problem is NP-hard, as shown in Theorem 1. Variations of the *PED* problem that use a fixed $max\mathcal{P} = 0$, or multiple thresholds, are easily dealt with by our algorithms.

Theorem 1 PED *is NP-hard.*

Proof. The proof is by reducing the NP-hard *Weighted Set Cover* (*WSC*) problem [12] to *PED*. The *WSC* problem is defined as follows. Given a collection $L = \{L_1, \ldots, L_m\}$ of sets, such that each $L_j \in L$ has a nonnegative weight $w(L_j)$ and $\cup_{L_j \in L} L_j = U = \{u_1, \ldots, u_n\}$, find a subcollection $L' \subseteq L$ that (I) covers all elements of U (i.e., $\cup_{L_j \in L'} L_j = U$), and (II) has minimum $\sum_{L_j \in L'} w(L_j)$.

We map a given instance $\mathcal{I}_{WSC}$ of *WSC* to an instance $\mathcal{I}_{PED}$ of *PED*, in polynomial time, as follows (see Fig. 4):

(I) Each subset $L_j \in L$ is mapped to $[s_j, x_j, (s_j, x_j)]$, where s_j is a seed, x_j is a non-seed node, and (s_j, x_j) is an edge.
(II) Each element u_i in each $L_j \in L$ is mapped to $[x_j, u_i, (x_j, u_i)]$, where u_i is a vulnerable node and (x_j, u_i) is an edge.
(III) We assign PRH to edges as follows: $PRH((s_j, x_j)) = w(L_j)$ and $PRH((x_j, u_i)) = \infty$, to force the algorithm for *PED* to select (s_j, x_j), which corresponds to L_j, and prevent the selection of (x_j, u_i).
(IV) We assign edge probabilities as follows: $p((s_j, x_j)) = 1$ and $p((x_j, u_i)) = \frac{1}{|n^-(u_i)|}$, to ensure that the path probability of $[(s_j, x_j), (x_j, u_i)]$ is determined by $|n^-(u_i)|$, which corresponds to the frequency of the element u_i over the subsets of L (number of subsets containing u_i).
(V) We set $max\mathcal{P} = 1 - \frac{1}{max_{u_i} |n^-(u_i)|}$, so that at least one path $[(s_j, x_j), (x_j, u_i)]$ to each u_i is disconnected after the deletion of the selected edges by the algorithm for *PED*. This corresponds to covering each element $u_i \in U$ by at least one subset L_j.

In the following, we prove the correspondence between a solution L' to the given instance $\mathcal{I}_{WSC}$ and a solution E' to the instance $\mathcal{I}_{PED}$.

We first prove that, if L' is a solution to $\mathcal{I}_{WSC}$, then E' is a solution to $\mathcal{I}_{PED}$. Since $\cup_{L_j \in L'} L_j = U = \{u_1, \ldots, u_n\}$, the deletion of E' disconnects at least one path to each u_i, $i \in [1, n]$, and $\mathcal{P}_{u_i} \leq max\mathcal{P}$ holds, for each u_i. Since $\sum_{L_j \in L'} w(L_j)$ is minimum, $PRH(E') = \sum_{L_j \in L'} w(L_j)$ is minimum. Thus, E' is a solution to $\mathcal{I}_{PED}$.

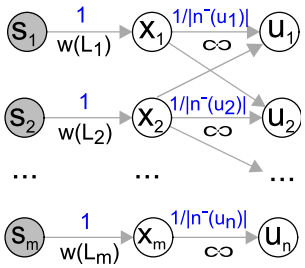


Fig. 4: The graph created from an instance of the *WSC* problem. The seeds are $s_1, \ldots, s_m$ and the vulnerable nodes are $u_1, \ldots, u_n$. The edge probability (resp. PRH) appears above (resp. below) the edges.

We now prove that, if an edge subset E' is a solution to $\mathcal{I}_{PED}$, then L' is a solution to $\mathcal{I}_{WSC}$. Since E' is a solution to $\mathcal{I}_{PED}$, at least one path to each u_i is disconnected, and $\mathcal{P}_{u_i} \leq max\mathcal{P}$ holds for each u_i, $i \in [1, n]$. Thus, L' satisfies $\cup_{L_j \in L'} L_j = \{u_1, \ldots, u_n\} = U$. Since $PRH(E') = \sum_{L_j \in L'} w(L_j)$ is minimum, L' has minimum $\sum_{L_j \in L'} w(L_j)$. Thus, L' is a solution to $\mathcal{I}_{WSC}$. □

5 Addressing *PED* for a single vulnerable node

This section details our methodology for addressing *PED*, when there is a single vulnerable node v. The main idea is to model *PED* as a *Submodular Set Cover* (*SSC*) [14, 41] problem and to develop an algorithm for *PED* based on the approximation algorithm for *SSC* [41]. Our algorithm is called GDE and is applied to the activation graph $\tilde{G}_v$ of v. The use of $\tilde{G}_v$ improves efficiency, since only edges that affect the activation probability of v are considered (see Section 2.3).

Modeling *PED* as *SSC*. We show that *PED*, for a single vulnerable node, can be modeled as an *SSC* problem, by means of a reduction. We first provide the definition of the *SSC* problem [14] and then a formulation of *PED* based on *SSC*, which is referred to as PED_{SSC} and is used in the reduction. After that, we present the reduction from PED_{SSC} to *SSC*.

Definition 1 (Submodular Set Cover (SSC) [14]) Let U be a universe of elements and $c(u)$ be the nonnegative cost of an element u of U. Let also C be a function defined as $C(\mathcal{S}) = \sum_{u \in \mathcal{S}} c(u)$, for a subset $\mathcal{S}$ of U, and F be a monotone non-decreasing submodular function. Given a nonnegative constant b, find a subset $\mathcal{S} \subseteq U$, so that the following two conditions hold:

(I) the cost $C(\mathcal{S})$ is minimum, and
(II) $F(\mathcal{S}) \geq b$.

The *PED*$_{\text{SSC}}$ problem is defined as follows.

Problem 2 (PED$_{\text{SSC}}$) Given a threshold $max\mathcal{P}$ in $[0,1]$, a set of seed nodes S, a vulnerable node v, the activation graph $\tilde{G}_v$, and the PRH of each edge of $\tilde{G}_v$, find an edge subset $E' \subseteq E$ of $\tilde{G}_v$, such that: *(I)* $PRH(E')$ is minimum, and *(II)* $\mathcal{P}(v, Q_{S,v}, E) - \mathcal{P}(v, Q_{S,v}, E') \leq max\mathcal{P}$, where $\mathcal{P}(v, Q_{S,v}, E)$ *(*resp., $\mathcal{P}(v, Q_{S,v}, E')$*)* is the activation probability of v by the paths of $Q_{S,v}$ *that contain edges in* E *(*resp., in E'*)*.

Both *SSC* and *PED*$_{\text{SSC}}$ are constrained optimization problems, in which the objective function (i.e., the function C in *SSC* and the function PRH in *PED*$_{\text{SSC}}$) is a monotone linear function. In addition, the constraint function, F, in *SSC* is monotone non-decreasing submodular.

As we will show later, the reduction requires to map an instance of *PED*$_{\text{SSC}}$ to an instance of *SSC*. This is possible when: (I) the PRH function is monotone and linear (as the function C in *SSC*), and (II) the constraint function, $\mathcal{P}(v, Q_{S,v}, E')$, in *PED*$_{\text{SSC}}$ is monotone non-decreasing submodular (as the function F in *SSC*). Clearly, PRH is monotone and linear. To show that $\mathcal{P}(v, Q_{S,v}, E')$ is monotone non-decreasing submodular, we provide Theorem 2. Intuitively, the submodularity property holds, because the addition of an edge e into E' increases $\mathcal{P}(v, Q_{S,v}, E')$ by the sum of the path probabilities of paths that contain e and no other edge in E'. Thus, the increase caused by adding e into E' is at least equal to the increase to $\mathcal{P}(v, Q_{S,v}, E'')$ caused by adding e into a superset E'' of E'.

Theorem 2 *The function $\mathcal{P}(v, Q_{S,v}, E')$ is monotone non-decreasing submodular.*

Proof: Let E_v be the edge set of $\tilde{G}_v$, $E_1 \subseteq E_2 \subseteq E_v$ be subsets of E_v, and e be an edge in $E_v \setminus E_2$. Let also $Q^{E_1}_{S,v} \subseteq Q_{S,v}$ and $Q^{E_2}_{S,v} \subseteq Q_{S,v}$ be the set of paths containing edges in E_1 and E_2, respectively, and $Q^{e}_{S,v} \subseteq Q_{S,v}$ be the set of paths containing e. We will show that Eq. 5 holds in each of the following cases.

$$\mathcal{P}(v, Q_{S,v}, E_1 \cup e) - \mathcal{P}(v, Q_{S,v}, E_1) \geq \mathcal{P}(v, Q_{S,v}, E_2 \cup e) - \mathcal{P}(v, Q_{S,v}, E_2) \quad (5)$$

Case I: All paths in $Q^{e}_{S,v}$ are contained in $Q^{E_1}_{S,v}$. Thus,

$$\mathcal{P}(v, Q_{S,v}, E_1 \cup e) - \mathcal{P}(v, Q_{S,v}, E_1) = 0 \geq \mathcal{P}(v, Q_{S,v}, E_2 \cup e) - \mathcal{P}(v, Q_{S,v}, E_2) = 0$$

since adding e does not change $Q^{E_1}_{S,v}$ and $Q^{E_2}_{S,v}$.

Case II: All paths in $Q^{e}_{S,v}$ are contained in $Q^{E_2}_{S,v}$ and at least one path is not contained in $Q^{E_1}_{S,v}$. Thus,

$$\mathcal{P}(v, Q_{S,v}, E_1 \cup e) - \mathcal{P}(v, Q_{S,v}, E_1) \geq \mathcal{P}(v, Q_{S,v}, E_2 \cup e) - \mathcal{P}(v, Q_{S,v}, E_2) = 0$$

since adding e adds paths into $Q^{E_1}_{S,v}$ only.

Case III: At least one path in $Q^{e}_{S,v}$ is not contained in $Q^{E_2}_{S,v}$. Thus,

$$\mathcal{P}(v, Q_{S,v}, E_1 \cup e) - \mathcal{P}(v, Q_{S,v}, E_1) \geq \mathcal{P}(v, Q_{S,v}, E_2 \cup e) - \mathcal{P}(v, Q_{S,v}, E_2)$$

since adding e adds into $Q^{E_1}_{S,v}$ all the paths that are added into $Q^{E_2}_{S,v}$ and the paths of $Q^{e}_{S,v}$ contained in $Q^{E_2}_{S,v} \setminus Q^{E_1}_{S,v}$.

Consequently, Eq. 5 holds in each case and $\mathcal{P}(v, Q_{S,v}, E')$ is submodular. In addition, $\mathcal{P}(v, Q_{S,v}, E')$ is monotone, since $\mathcal{P}(v, Q_{S,v}, E') \leq \mathcal{P}(v, Q_{S,v}, E' \cup e)$ for each $e \notin E'$, and non-decreasing, since $\mathcal{P}(v, Q_{S,v}, E_1) \leq \mathcal{P}(v, Q_{S,v}, E_2)$. □

We now present the reduction from *PED*$_{\text{SSC}}$ to *SSC*.

Theorem 3 PED*$_{SSC}$ can be reduced to* SSC.

Proof. We provide a reduction from *PED*$_{\text{SSC}}$ to *SSC*, by defining a pair of polynomial-time computable functions (f, g), such that: (I) f maps any given instance $\mathcal{I}_{PED_{SSC}}$ of *PED*$_{\text{SSC}}$ to an instance $\mathcal{I}_{SSC}$ of *SSC*, and (II) g maps any feasible solution $\mathcal{S}$ of $\mathcal{I}_{SSC}$ to back to a feasible solution E' of $\mathcal{I}_{PED_{SSC}}$, while preserving the approximation ratio (i.e., the approximation ratio of $\mathcal{S}$ with respect to $\mathcal{I}_{SSC}$ is the same as that of E' with respect to $\mathcal{I}_{PED_{SSC}}$).

The function f gets as input any instance $\mathcal{I}_{PED_{SSC}}$ of *PED*$_{\text{SSC}}$ and constructs an instance $\mathcal{I}_{SSC}$ of *SSC*, in polynomial time, as follows: (I) for each edge e with $PRH(e)$ in the activation graph $\tilde{G}_v$, it adds into the universe U an element u with cost $c(u) = PRH(e)$, (II) it defines the function $F(\mathcal{S}) = \mathcal{P}(v, Q_{S,v}, E')$, where E' is the edge subset corresponding to $\mathcal{S} \subseteq U$, and (III) it sets $b = \mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P}$. The function g gets as input any feasible solution $\mathcal{S}$ of $\mathcal{I}_{SSC}$ and constructs a feasible solution E' of $\mathcal{I}_{PED_{SSC}}$, in polynomial time, by adding into E' the edges that correspond to the elements of $\mathcal{S}$. Note that E' is a solution of $\mathcal{I}_{PED_{SSC}}$. This is because $F(\mathcal{S}) \geq b$ implies $\mathcal{P}(v, Q_{S,v}, E') \geq \mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P}$, which implies $\mathcal{P}(v, Q_{S,v}, E) - \mathcal{P}(v, Q_{S,v}, E') \leq max\mathcal{P}$. In addition, $PRH(E') = C(\mathcal{S})$, which implies that the approximation ratio of $\mathcal{S}$ with respect to $\mathcal{I}_{SSC}$ is the same as that of E' with respect to $\mathcal{I}_{PED_{SSC}}$. □

Greedy Delete Edges (GDE). Since *PED*$_{\text{SSC}}$ can be modeled as an *SSC* problem, we can obtain an approximate solution to *PED*$_{\text{SSC}}$ using the algorithm of [41]. This algorithm iteratively adds the element $u \in U \setminus \mathcal{S}$ with the minimum ratio $\frac{c(u)}{F(\mathcal{S} \cup u) - F(\mathcal{S})}$ into $\mathcal{S}$, until $F(\mathcal{S}) \geq b$. That is, we can create an instance of *SSC* from a given instance of *PED*$_{\text{SSC}}$, use the algorithm of [41] to obtain a solution to the instance of *SSC* and then map back the solution to a solution of *PED*$_{\text{SSC}}$, as in the proof of Theorem 3. We do not describe this process in detail, for clarity. Instead, in the following, we write

the GDE algorithm "around" the algorithm of [41]. That is, GDE employs the same element selection and stopping criterion as the algorithm of [41] but uses the terminology for PED_{SSC}.

Algorithm: GDE
Input: Graph G, activation graph $\tilde{G}_v$, threshold $max\mathcal{P}$, PageRank distribution $PR(G)$, restart probability α
Output: Set of edges E'
1 **foreach** *edge* $e = (u, u')$ *of* $\tilde{G}_v$ **do**
2 $\quad PRH(e) \leftarrow (1-\alpha) \cdot \frac{PR(u)}{|n^+(u)|}$
3 $E' \leftarrow \varnothing$
4 **while** $\mathcal{P}(v, Q_{S,v}, E) - \mathcal{P}(v, Q_{S,v}, E') > max\mathcal{P}$ **do**
5 $\quad$ Reconstruct $\tilde{G}_v$ and find an edge e of $\tilde{G}_v$ s.t. $g_{\mathcal{P}}(e) > 0$ and $\frac{PRH(e)}{g_{\mathcal{P}}(e)}$ is minimum
6 $\quad E' \leftarrow E' \cup e$
7 Delete E' from G
8 **return** E'

GDE is applied to the activation graph $\tilde{G}_v$ of v and constructs the subset of edges E' to be deleted iteratively. As can be seen in the pseudocode, the algorithm computes the PRH of each edge in $\tilde{G}_v$ (steps 1-2) and constructs E', based on a similar criterion to that of the algorithm of [41] (steps 4-6). That is, it selects the edge e with the minimum ratio $\frac{PRH(e)}{g_{\mathcal{P}}(e)}$, where $g_{\mathcal{P}}(e) = \mathcal{P}(v, Q_{S,v}, E' \cup e) - \mathcal{P}(v, Q_{S,v}, E')$ is the *path probability gain* of e. The path probability gain $g_{\mathcal{P}}(e)$ quantifies the increase in $\mathcal{P}(v, Q_{S,v}, E')$, caused by the selection of e. Thus, the selected edge has small PRH and contributes significantly to lowering the activation probability $\mathcal{P}_v$. To ensure that $g_{\mathcal{P}}(e)$ is positive, we reconstruct $\tilde{G}_v$ in step 5. Next, e is added into E' (step 6), and the process is repeated if the activation probability $\mathcal{P}(v, Q_{S,v}, E) - \mathcal{P}(v, Q_{S,v}, E')$ exceeds $max\mathcal{P}$. Last, E' is deleted from G and returned (steps 7-8).

Theorem 4 shows that GDE finds a solution with PRH at most $1 + ln(\lambda)$ times larger than that of the optimal solution, where λ depends on the PRH and path probability gain of the selected edges. The proof easily follows from [41] (omitted).

Theorem 4 *Let E' be the output of* GDE *and E'_{OPT} be the optimal solution to* PED_{SSC}. *It holds that $PRH(E') \leq (1 + ln(\lambda)) \cdot PRH(E'_{OPT})$, where λ is the minimum of:* (I) *the maximum ratio $\frac{g_{\mathcal{P}}(e_1)}{g_{\mathcal{P}}(e)}$,* (II) $\frac{PRH(e_\ell)}{g_{\mathcal{P}}(e_\ell)} / \frac{PRH(e_1)}{g_{\mathcal{P}}(e_1)}$, *and* (III) $\frac{\mathcal{P}(v, Q_{S,v}, E')}{g_{\mathcal{P}}(e_\ell)}$, *where e_1 (resp., e_ℓ) is the edge that was first (resp., last) added into E', and e is an edge in $E' \setminus e_1$.*

GDE needs $O(|E_v| \cdot |E'| \cdot T)$ time, where E_v is the edge set of $\tilde{G}_v$, $E' \subseteq E_v$ is the set of deleted edges, and T is the maximum time needed to compute $g_{\mathcal{P}}(e)$. Specifically, step 4 is executed $O(|E'|)$ times, and step 5 needs $O(|E_v| \cdot T)$ time. In practice, $T << |E_v|$ because the activation probabilities are computed using small subgraphs of $\tilde{G}_v$ (see Section 2.3).

6 Algorithms for multiple vulnerable nodes

This section presents AGDE and IGDE, which address the *PED* problem when there are multiple vulnerable nodes.

Aggregate Greedy Delete Edges (AGDE). AGDE is an approximation algorithm, which reduces the activation probabilities of multiple vulnerable nodes simultaneously. The main idea is to model *PED* as an *SSC* problem and to base AGDE on the algorithm of [41]. However, the condition (II) of *PED* (i.e., $\mathcal{P}_v \leq max\mathcal{P}$ after the deletion of E', for each vulnerable node v) involves multiple constraints, whereas in *SSC* there is a single constraint, $F(\mathcal{S}) \geq b$ (see Definition 1). Therefore, to model *PED* as *SSC*, we need to replace the multiple constraints of *PED* with a single, aggregate constraint which is equivalent to them (i.e., the aggregate constraint is satisfied if and only if all constraints in the condition (II) of *PED* are satisfied).

To formulate the aggregate constraint, we observe that:

(I) The constraint for each vulnerable node v in the condition (II) of *PED* can be written as $\mathcal{P}(v, Q_{S,v}, E) - \mathcal{P}(v, Q_{S,v}, E') \leq max\mathcal{P}$, or equivalently as

$$\mathcal{P}(v, Q_{S,v}, E') \geq \mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P} \tag{6}$$

(II) The constraint in Eq. 6 can be written as

$$min(\mathcal{P}(v, Q_{S,v}, E'), \mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P}) = \mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P} \tag{7}$$

To see this, note that the min in Eq. 7 reduces (truncates) $\mathcal{P}(v, Q_{S,v}, E')$ to $\mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P}$, if and only if $\mathcal{P}(v, Q_{S,v}, E') \geq \mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P}$ is satisfied. Thus, we can check whether the constraint in Eq. 6 is satisfied, by checking whether the value of min in Eq. 7 is equal to the constant $\mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P}$. This allows to aggregate the constraints of all vulnerable nodes into a single constraint, as shown in step (III) below.

(III) All constraints $\mathcal{P}(v, Q_{S,v}, E') \geq \mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P}$ in the condition (II) of *PED* can be replaced by the aggregate constraint

$$\sum_v min(\mathcal{P}(v, Q_{S,v}, E'), \mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P}) = \sum_v (\mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P}) \tag{8}$$

To see this, note that the constraint in Eq. 8 is satisfied, if and only if each constraint in Eq. 7 is satisfied, which implies that all constraints in the condition (II) of *PED* are satisfied.

We now present a formulation of the *PED* problem, which uses the aggregate constraint in Eq. 8. For clarity, the problem in this formulation is referred to as *PED*$_{Aggr}$ and the term $\sum_v min(\mathcal{P}(v, Q_{S,v}, E'), \mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P})$ in Eq. 8 is denoted with $\mathcal{P}(D, \cup_{v\in D} Q_{S,v}, E')$ and referred to as *aggregate path probability*.

Problem 3 (PED$_{Aggr}$) Let $S \subseteq V$ be the subset of seed nodes, $D \subseteq V$ be the subset of vulnerable nodes, and $\tilde{G}_D = \cup_{v\in D}\tilde{G}_v$ be the activation graph of D. Given a threshold $max\mathcal{P}$ in $[0, 1]$ and the PRH of each edge of $\tilde{G}_D$, find an edge subset $E' \subseteq E$ of $\tilde{G}_D$ such that:

(I) $PRH(E')$ is minimum, and
(II) $\mathcal{P}(D, \cup_{v\in D} Q_{S,v}, E') = \sum_{v\in D} (\mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P})$.

Note that *PED*$_{Aggr}$ contains a single constraint and that the aggregate path probability $\mathcal{P}(D, \cup_{v\in D} Q_{S,v}, E')$ is submodular (this easily follows from the submodularity of $\mathcal{P}(v, Q_{S,v}, E')$). Thus, *PED*$_{Aggr}$ can be reduced to *SSC* (the reduction is omitted because it is similar to that of Theorem 3) and be approximated by using the algorithm of [41] as the basis of our AGDE algorithm.

In what follows, we present the AGDE algorithm. As can be seen in the pseudocode, the algorithm is applied to the activation graph $\tilde{G}_D$ and iteratively selects the edge with the minimum ratio $\frac{PRH(e)}{g_{\mathcal{P}_D}(e)}$ (step 5), where $g_{\mathcal{P}_D}(e)$ is the *aggregate path probability gain*, defined as $\mathcal{P}(D, \cup_{v\in D} Q_{S,v}, E' \cup e) - \mathcal{P}(D, \cup_{v\in D} Q_{S,v}, E')$. The process is repeated until the condition (II) of *PED*$_{Aggr}$ holds. Note that this condition holds, in the worst case when E' contains all edges of $\tilde{G}_D$. Thus, AGDE will always terminate.

Algorithm: AGDE
Input: Graph G, activation graph $\tilde{G}_D$, threshold $max\mathcal{P}$, set of vulnerable nodes D, PageRank distribution $PR(G)$, restart probability α
Output: Set of edges E'
1 **foreach** *edge* $e = (u, u')$ *of* $\tilde{G}_D$ **do**
2 $\quad PRH(e) \leftarrow (1 - \alpha) \cdot \frac{PR(u,G)}{|n^+(u)|}$
3 $E' \leftarrow \varnothing$
4 **while** $\mathcal{P}(D, \cup_{v\in D} Q_{S,v}, E') < \sum_{v\in D}(\mathcal{P}(v, Q_{S,v}, E) - max\mathcal{P})$ **do**
5 $\quad$ Reconstruct $\tilde{G}_D$ and find an edge e of $\tilde{G}_D$ s.t. $g_{\mathcal{P}_D}(e) > 0$ and $\frac{PRH(e)}{g_{\mathcal{P}_D}(e)}$ is minimum
6 $\quad E' \leftarrow E' \cup e$
7 Delete E' from G
8 **return** E'

Theorem 5 shows that AGDE finds a solution with PRH at most $1 + ln(\lambda_D)$ times larger than that of the optimal solution to *PED*$_{Aggr}$, where λ_D depends on the PRH and aggregate path probability gain of the selected edges. The proof easily follows from [41] (omitted).

Theorem 5 *Let E' be the output of* AGDE *and E'_{OPT} be the optimal solution to* PED$_{Aggr}$. *It holds that $PRH(E') \leq (1 + ln(\lambda_D)) \cdot PRH(E'_{OPT})$, where λ_D is the minimum of:* (I) *the maximum ratio $\frac{g_{\mathcal{P}_D}(e_1)}{g_{\mathcal{P}_D}(e)}$,* (II) $\frac{PRH(e_\ell)}{g_{\mathcal{P}_D}(e_\ell)} / \frac{PRH(e_1)}{g_{\mathcal{P}_D}(e_1)}$, *and* (III) $\frac{\mathcal{P}(D, \cup_{v\in D} Q_{S,v}, E')}{g_{\mathcal{P}_D}(e_\ell)}$, *where e_1 (resp., e_ℓ) is the edge that was first* (*resp., last*) *added into E', and e is an edge in $E' \setminus e_1$.*

Clearly, AGDE needs $O(|E| \cdot |E'| \cdot T_D)$ time, where E is the edge set of $\tilde{G}_D$, $E' \subseteq E$ is the set of deleted edges, and T_D is the maximum time needed to compute $g_{\mathcal{P}_D}$.

Iterative Greedy Delete Edges (IGDE). As can be seen in the pseudocode, IGDE sorts the vulnerable nodes, in decreasing order of activation probability, and applies GDE to the activation graph $\tilde{G}_v$ of one vulnerable node v at a time.

This heuristic improves efficiency, because: (I) $\tilde{G}_v$ contains a much smaller number of edges than the activation graph of all vulnerable nodes to which AGDE is applied, and (II) the edge subset E'_v that is deleted in an iteration is not considered again. However, this reduces the number of explored solutions. Therefore, vulnerable nodes with large activation probability $\mathcal{P}(v, Q_{S,v}, E)$ are dealt with first, when more edges are available for deletion.

Algorithm: IGDE
Input: Graph G, threshold $max\mathcal{P}$, set of vulnerable nodes D, activation graph $\tilde{G}_v$ for each $v \in D$, PageRank distribution $PR(G)$, restart probability α
Output: Set of edges E'
1 sort each v in D in decreasing order of activation probability $\mathcal{P}(v, Q_{S,v}, E)$
2 $E' \leftarrow \varnothing$; $G_{tmp} \leftarrow G$
3 **foreach** v *in* D **do**
4 $\quad$ **if** $\mathcal{P}(v, Q_{S,v}, E) - \mathcal{P}(v, Q_{S,v}, E') > max\mathcal{P}$ **then**
5 $\quad\quad E'_v \leftarrow$ GDE(G, $\tilde{G}_v, max\mathcal{P}, PR(G_{tmp}), \alpha)$
6 $\quad\quad$ **foreach** v *in* D **do**
7 $\quad\quad\quad$ Update $\tilde{G}_v$ to reflect the deletion of E'_v
8 $\quad\quad E' \leftarrow E' \cup E'_v$
9 **return** E'

Note that each vulnerable node v is considered once, because $\mathcal{P}(v, Q_{S,v}, E')$ cannot decrease in the next iterations (see Theorem 2). Thus, after the loop of step 3 terminates, the condition (II) of *PED* holds, and the subset of edges E' is returned (step 9). Furthermore, GDE is applied to the PageRank distribution of the original graph (step 5), so that edge deletion does not affect the PRH computation in GDE.

IGDE needs $O(\sum_{v\in D}(|E_v| \cdot |E'| \cdot T + |D| \cdot |E_v|))$ time. Specifically, step 5 takes $O(|E_v| \cdot |E'| \cdot T)$ time, while steps 6 and 7 take $O(|D| \cdot |E_v|)$ time.

7 Efficiency optimization using lazy edge selection

In this section, we propose a lazy edge selection technique that is used to improve the efficiency of our algorithms, without affecting the quality of their solutions. The versions of GDE, AGDE, and IGDE that implement the technique are referred to as GDE_{LAZY}, $\text{AGDE}_{\text{LAZY}}$, and $\text{IGDE}_{\text{LAZY}}$, respectively. The technique is based on the submodularity of the functions $\mathcal{P}(v, Q_{S,v}, E')$ and $\mathcal{P}(D, \cup_{v \in D} Q_{S,v}, E')$, which are used in the edge selection criteria of our algorithms. Our technique is inspired by *Accelerated Greedy* (also referred to as Lazy Greedy) algorithm [34], which was reviewed in Section 2.4.

Note that there are other algorithms [16, 35] that can be used instead of Accelerated Greedy, in order to find a subset $U' \subseteq U$ with approximately maximum $f(U')$ and size r. However, our lazy edge selection technique is not based on them. This is because they require efficiently computing the gain of two element subsets simultaneously [16], which is not feasible in our case, or because they cannot maintain the approximation ratio of our algorithms [35], since they are designed for the submodular maximization problem [36].

The lazy edge selection technique maintains a priority queue, which stores the ratio between PRH and the gain $g_{\mathcal{P}}$ for each edge in the case of GDE_{LAZY} and $\text{IGDE}_{\text{LAZY}}$ (respectively, $g_{\mathcal{P}_D}$ in the case of $\text{AGDE}_{\text{LAZY}}$), and it is sorted in increasing order. In the first iteration, the gains are computed with $E' = \varnothing$ and the top entry is retrieved from the priority queue. The top entry corresponds to the edge with the minimum ratio (i.e., the "best" edge in the current iteration). Then, the edge corresponding to the top entry is added into E', and the top entry is removed from the queue. In each subsequent iteration, the top entry of the queue is retrieved and its ratio is updated to reflect the current E'. If the top entry remains on the top of the queue after the update, it is removed from the queue and its corresponding edge e is added into E'. Otherwise, the current top entry is updated and the process is repeated. Thus, lazy edge selection is similar to *Accelerated Greedy* in that it exploits the submodularity of the gain to avoid updating all entries of the priority queue.

In the following, we explain how the lazy edge selection technique exploits the submodularity of the gain to avoid updating all entries of the priority queue. Let E'_i be the set of selected edges by the lazy edge selection strategy in iteration i, g_i be the gain for an edge in iteration i (e.g., the gain $g_{\mathcal{P}}$ in the case of GDE_{LAZY}), and $e \notin E'_i$ be an edge whose corresponding entry remained on the top of the queue after the update in the next iteration $i+1$. Since the entry corresponding to e was on the top of the priority queue before the update, $\frac{PRH(E'_i \cup \{e\})}{g_i(e)} \leq \frac{PRH(E'_i \cup \{e'\})}{g_i(e')}$, for each edge $e' \notin E'_i \cup \{e\}$. That is, in iteration i, e was the best available edge to select, except the selected edges in E'_i. Since the entry corresponding to the edge e remained on the top of the priority queue after the update, it holds that

$$\frac{PRH(E'_{i+1} \cup \{e\})}{g_{i+1}(e)} \leq \frac{PRH(E'_i \cup \{e'\})}{g_i(e')}, \tag{9}$$

for each edge $e' \notin E'_i \cup \{e\}$. Note also that $PRH(E'_i \cup \{e'\}) \leq PRH(E'_{i+1} \cup \{e'\})$, due to the monotonicity of PRH, and $g_i(e') \geq g_{i+1}(e')$, due to the submodularity of the gain function. Thus, $\frac{PRH(E'_i \cup \{e'\})}{g_i(e')} \leq \frac{PRH(E'_{i+1} \cup \{e'\})}{g_{i+1}(e')}$, and from Eq. 9 we obtain $\frac{PRH(E'_{i+1} \cup \{e\})}{g_i(e)} \leq \frac{PRH(E'_{i+1} \cup \{e'\})}{g_{i+1}(e')}$, for each edge $e' \notin E'_i \cup \{e\}$. That is, e is the best edge to select in iteration $i+1$, when the entry corresponding to e remains on the top of the queue after the update. Thus, any edge e' would not be selected and its corresponding entry does not need updating.

In practice, lazy edge improves efficiency substantially, since there are many entries that do not need updating and the updating is costly, while it does not affect quality, since the same edges are selected by the algorithms, irrespectively of whether lazy edge selection is employed.

Algorithm: GDE_{LAZY}
Input: Graph G, activation graph $\tilde{G}_v$, threshold $max\mathcal{P}$, PageRank distribution $PR(G)$, restart probability α
Output: Set of edges E'

1 $E' \leftarrow \varnothing$
2 $Q \leftarrow$ empty priority queue sorted in increasing order
3 **foreach** *edge* $e = (u, u')$ *of* $\tilde{G}_v$ **do**
4 $\quad PRH(e) \leftarrow (1-\alpha) \cdot \frac{PR(u)}{|n^+(u)|}$
5 $\quad Q \leftarrow \frac{PRH(e)}{g_{\mathcal{P}}(e)}$
6 **while** $\mathcal{P}(v, Q_{S,v}, E) - \mathcal{P}(v, Q_{S,v}, E') > max\mathcal{P}$ **do**
7 $\quad$ Reconstruct $\tilde{G}_v$
8 $\quad$ **if** $E' = \varnothing$ **then**
9 $\quad\quad e \leftarrow$ the edge corresponding to the top entry of Q
10 $\quad$ **else**
11 $\quad\quad$ **do**
12 $\quad\quad\quad$ Update the top entry of Q using the current E'
13 $\quad\quad$ **while** *the top entry of* Q *changes*
14 $\quad\quad e \leftarrow$ the edge corresponding to the top entry of Q
15 $\quad$ Remove the top entry of Q
16 $\quad E' \leftarrow E' \cup e$
17 Delete E' from G
18 **return** E'

As can be seen in the pseudocode of GDE_{LAZY}, the algorithm starts by initializing the priority queue with the ratio between PRH and the gain $g_{\mathcal{P}}$ of each edge, which is computed with $E' = \varnothing$ (steps 1 to 5). Then, GDE_{LAZY} iteratively selects the edge e with the minimum gain, as long as the activation probability $\mathcal{P}(v, Q_{S,v}, E) - \mathcal{P}(v, Q_{S,v}, E')$ exceeds $max\mathcal{P}$ (steps 6 to 14). In the first iteration, e corresponds to the top entry of the priority queue (steps 8 to 9). In any subsequent iteration, the edge corresponds to the top

entry of the priority queue, after updating the queue using the current E' (steps 10 to 14). Next, the algorithm removes the top entry of the priority queue and adds the edge e into E' (steps 15 to 16). Since GDE_{LAZY} selects the edge with the minimum ratio in each iteration, it produces a solution with the same PRH as GDE.

The pseudocode of $\text{AGDE}_{\text{LAZY}}$ and $\text{IGDE}_{\text{LAZY}}$ can be derived similarly (omitted).

8 Experimental evaluation

In this section, we evaluate our algorithms in terms of their effectiveness and efficiency. Since existing methods are not applicable to the *PED* problem, we compared our algorithms against baselines that use different edge selection criteria, and against the optimal, exhaustive method, BRUTEFORCE, which examines all edge subsets. In addition, we show that PRH is an effective and efficient heuristic to capture the change to the PageRank scores, caused by edge deletion.

Setup and datasets. To quantify the impact of edge deletion, we used: (I) PRH, (II) the L_1 distance, (III) the percentage of deleted edges, and (IV) the Kendall τ_b correlation ($K\tau_b$). $K\tau_b$ captures changes to the ranking of all nodes, with respect to their PageRank scores [5]. A $K\tau_b$ value of 1 implies no change to the ranking and larger values are preferred.

We implemented all algorithms in C++ and applied them to the following real datasets: *cit-HepPh* (*Ph*), *Wiki-vote* (*Wiki*), and *Polblogs* (*Pol*). The *Ph* dataset is available at `http://snap.stanford.edu/data` and represents a High Energy Physics citation graph. Each node u in the dataset corresponds to a paper and each edge (u, u') represents that the paper corresponding to the node u cited the paper corresponding to the node u'. The *Wiki* dataset is available at `http://snap.stanford.edu/data` and represents a whom-trusts-whom graph from Wikipedia. Each node u in the dataset represents a Wikipedia user, and each edge (u, u') represents that the user corresponding to the node u voted for the promotion of the user corresponding to the node u' to become an administrator. The *Pol* dataset is available at `http://www-personal.umich.edu/~mejn/` and represents a graph of weblogs on US politics. Each node u in the dataset represents a weblog, each edge (u, u') represents a hyperlink from the weblog corresponding to node u to the weblog corresponding to node u'. We also used two synthetic datasets, *AB* and *ER*, which were generated by the Albert-Barabasi and the Erdös-Rényi model, respectively. These models generate random graphs with different structural properties that are common in real networks (e.g., power-law degree distribution for the Albert-Barabasi model, and small diameter for the Erdös-Rényi model). This allows us to test our approach in a wider range of settings. Of note, random graphs generated by the Albert-Barabasi and the Erdös-Rényi model were used in other works on influence diffusion optimization [11, 25]. Table 1 summarizes the characteristics of each dataset and its default values for $max\mathcal{P}$, $|S|$ (# of seeds), and $|D|$ (# of vulnerable nodes). BRUTEFORCE does not scale to real datasets. Thus, it was applied to 1000 datasets, which have 16 nodes and 28 edges on average and were generated by the Erdös-Rényi model.

Dataset	$\|V\|$	$\|E\|$	(avg, max) in-deg.	$max\mathcal{P}$	$\|S\|$	$\|D\|$
Ph	34546	421578	(24.3, 846)	0.1	200	50
Wiki	7115	103689	(13.7, 452)	0.1	75	20
Pol	1490	19090	(11.9, 305)	0.1	500	20
AB	111150	500000	(9, 99907)	0.01	500	5
ER	5000	49917	(9.98, 24)	0.01	50	20

Table 1: Characteristics of datasets and default values for threshold $max\mathcal{P}$, number of seeds $|S|$, and number of vulnerable nodes $|D|$.

All edge probabilities were assigned by the *uniform* method (i.e., each incoming edge to u has edge probability $\frac{1}{|n^-(u)|}$) [11, 24] and the threshold h was set to 10^{-3}, as in [17] (recall from Section 2.3 that $\mathcal{P}_u$ is estimated based on paths with path probability at least equal to h). The vulnerable nodes were: (I) selected randomly among the top-10% of nodes with the largest in-degree, in *Ph*, *Wiki*, *Pol*, and *ER*, and (II) the 5 nodes with the largest in-degree, in all other datasets. This excludes nodes that are easy to deal with. To find the seeds, we considered each vulnerable node v and iteratively selected random paths of length at least 2 that end at v, until $\mathcal{P}_v \geq min(r \cdot max\mathcal{P}, 1)$, where $r \geq 1$ is a random integer. The start nodes of these paths were used as seeds. Since there were many other paths from seeds to vulnerable nodes, the activation graphs were large. All experiments ran on an Intel Xeon at 2.4Ghz with 12Gb RAM.

Quality of approximation. We demonstrate that AGDE finds near-optimal solutions, by comparing it to BRUTEFORCE. Fig. 5a shows the ratio between the PRH for AGDE and for BRUTEFORCE, as well as the approximation ratio $1 + ln(\lambda_D)$, when $max\mathcal{P} = 0.2$, for all 1000 datasets (sorted in decreasing PRH). The ratio is 1 for 70% of the datasets, 1.04 on average, and at most 1.7. The approximation ratio is 2.6 on average and at most 6. Thus, AGDE produced solutions that are close to optimal, and the ratio of AGDE to BRUTEFORCE was much lower than the approximation ratio. We omit the results for $\text{AGDE}_{\text{LAZY}}$, since it produced the same solutions with AGDE (recall that the lazy edge selection technique does not affect the quality of solutions).

Effectiveness. We demonstrate that AGDE and IGDE do not substantially affect the information propagation properties of the graph and that they delete a small number of edges. We compared our methods with two baselines: (I)

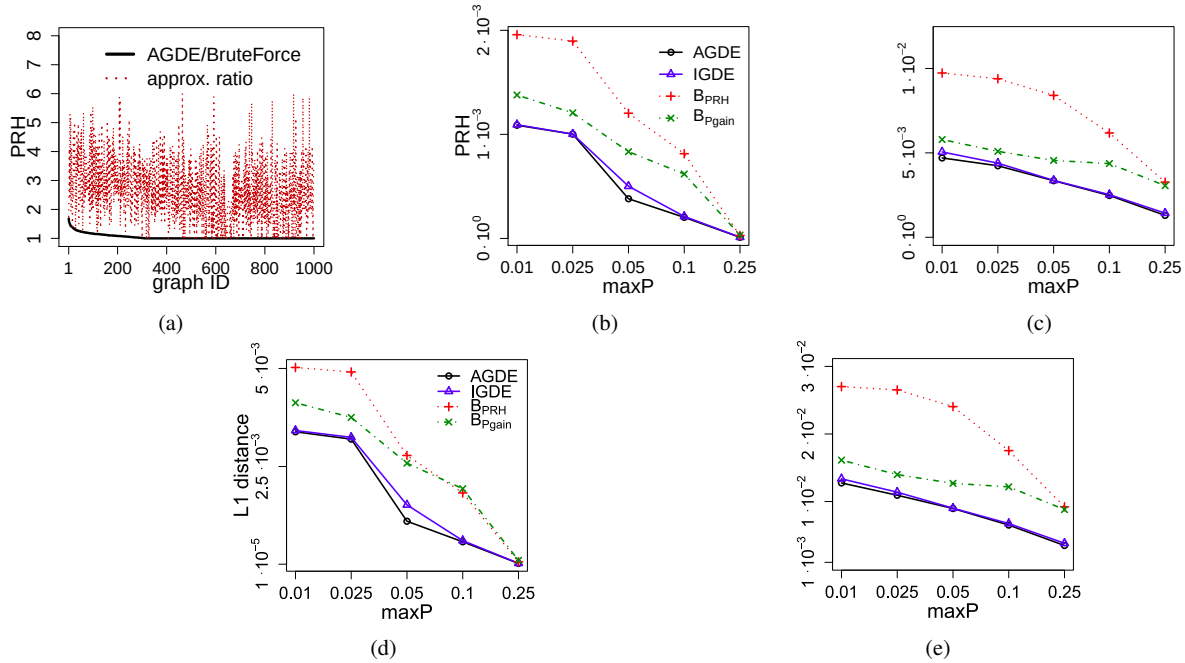


Fig. 5: (a) AGDE/BRUTEFORCE with respect to PRH and approximation ratio of AGDE for different random graphs. AGDE/BRUTEFORCE is defined as the ratio between the PRH of the solution generated by AGDE and the optimal solution generated by BRUTEFORCE. The approximation ratio of AGDE is defined as $1 + ln(\lambda_D)$ (see Section 6). (b) PRH vs. threshold $max\mathcal{P}$, for the *Wiki* dataset. (c) PRH vs. threshold $max\mathcal{P}$, for the *Pol* dataset. (d) L_1 distance vs. threshold $max\mathcal{P}$, for the *Wiki* dataset. (e) L_1 distance vs. threshold $max\mathcal{P}$, for the *Pol* dataset.

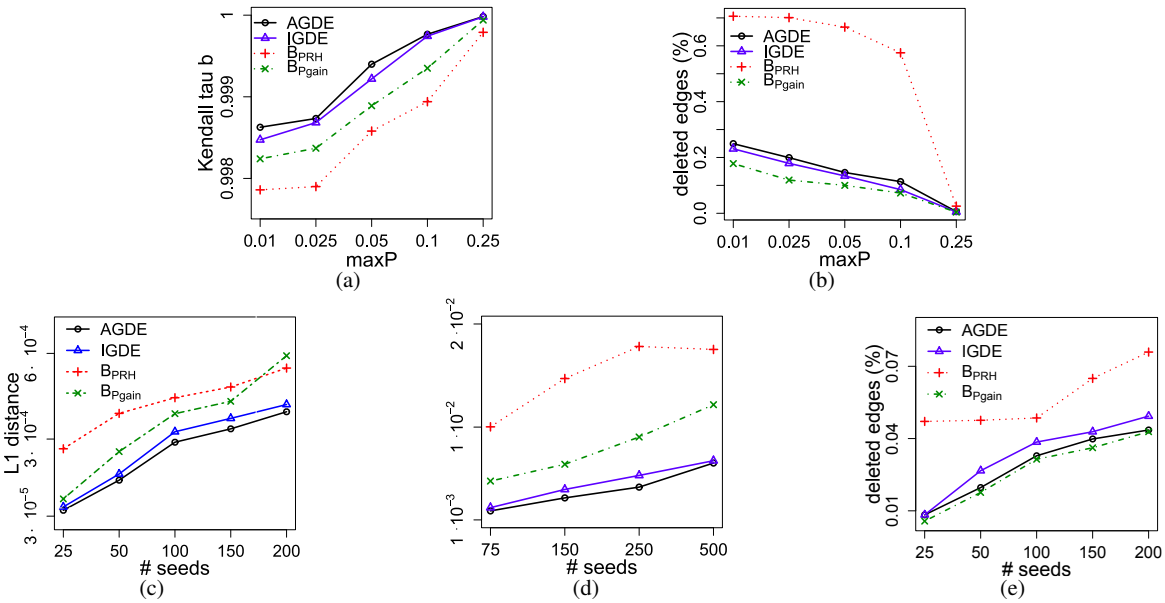


Fig. 6: (a) Kendall τ_b correlation ($K\tau_b$) vs. threshold $max\mathcal{P}$, for the *Wiki* dataset. $K\tau_b$ captures changes to the ranking of all nodes, with respect to their PageRank scores, and larger $K\tau_b$ scores imply smaller changes to the ranking. (b) Percentage of deleted edges vs. threshold $max\mathcal{P}$, for the *Wiki* dataset. (c) L_1 distance vs. number of seeds $|S|$, for the *Ph* dataset. (d) L_1 distance vs. number of seeds $|S|$, for the *Pol* dataset. (e) Percentage of deleted edges vs. number of seeds $|S|$, for the *Ph* dataset.

B$_{PRH}$, which selects the edge with the minimum PRH, and (II) B$_{PGain}$, which selects the edge with the maximum

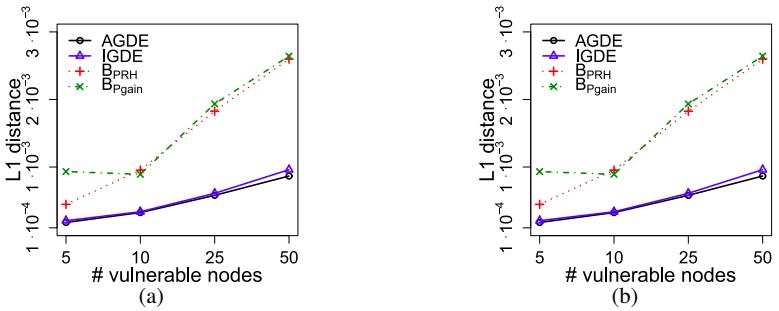


Fig. 7: (a) L_1 distance vs. number of vulnerable nodes $|D|$, for the *Wiki* dataset. (b) Percentage of deleted edges vs. number of vulnerable nodes $|D|$, for the *Wiki* dataset.

aggregate path probability gain. Both baselines are based on AGDE but do not offer approximation guarantees. We omit the results for AGDE$_{\text{LAZY}}$ and IGDE$_{\text{LAZY}}$, since they produced the same solutions with AGDE and IGDE, respectively.

Figs. 5b and 5c show the PRH, for varying $max\mathcal{P}$. The PRH decreases as $max\mathcal{P}$ increases, because the required reduction in activation probabilities becomes smaller. AGDE was the best method, and the PRH for IGDE was slightly larger. The baselines performed much worse, because B$_{PRH}$ deleted edges that did not reduce the activation probabilities of vulnerable nodes and B$_{PGain}$ deleted edges with large PRH.

Figs. 5d and 5e show the results for the L_1 distance, which follows the same trend as PRH. This suggests that minimizing PRH helps preserving the PageRank distribution. AGDE and IGDE performed similarly with respect to $K\tau_b$ and better than the baselines (see Fig. 6a). Furthermore, AGDE and IGDE deleted at most 0.04% more edges than B$_{PGain}$, which aims to minimize the number of deleted edges (see Fig. 6b).

Next, we measured effectiveness, for varying $|S|$, using seed sets of increasing size, whose elements were contained in all larger sets. Figs. 6c and 6d show that the L_1 distance increases with $|S|$. This is because the activation probabilities of vulnerable nodes, before edge deletion, are higher for large seed sets. They also show that AGDE and IGDE outperformed both baselines. Furthermore, AGDE and IGDE deleted at most 0.01% more edges than B$_{PGain}$ (see Fig. 6e).

We also measured effectiveness, for varying $|D|$ (# of vulnerable nodes). AGDE and IGDE performed similarly and significantly better than both baselines, with respect to the L_1 distance (see Fig. 7a). Furthermore, our methods deleted at most 0.5% more edges than B$_{PGain}$ (see Fig. 7b).

Thus, AGDE and IGDE preserved the information propagation properties of the graph much better than both baselines, and they deleted a similar number of edges with the B$_{PGain}$ baseline, which aims to minimize the number of deleted edges.

Efficiency comparison with baselines. We demonstrate that AGDE and IGDE scale well with $|S|$, $|D|$, and $|E|$, and that they are more efficient than the fastest baselines, B$_{PRH}$ and B$_{PGain}$. In addition, we show that IGDE is substantially more efficient than AGDE.

Fig. 8a shows that AGDE and IGDE scaled better than linear (*sublinearly*) with $|S|$. However, IGDE was up to 4 times faster, as it considers seeds contained in the activation graph of one vulnerable node at a time. Fig. 8b shows that IGDE scaled sublinearly with $|D|$, and it was up to *two orders of magnitude faster* than AGDE. This is because the edges deleted in an iteration of IGDE affected many activation graphs. Fig. 8c shows that AGDE and IGDE scaled sublinearly with $|E|$, and that IGDE was up to one order of magnitude faster. The baselines scaled similarly to AGDE, and the results for the *ER* dataset were similar (omitted).

Efficiency benefit of lazy edge selection. We demonstrate that the lazy edge selection strategy substantially improves the efficiency of both AGDE and IGDE. To show the improvement, we report: (I) the runtime of AGDE$_{\text{LAZY}}$ and IGDE$_{\text{LAZY}}$, and (II) the *Average savings* brought by lazy edge selection. For AGDE$_{\text{LAZY}}$, the *Average savings* measure is computed as the percentage of edges of the activation graph $\tilde{G}_D$ whose ratio is not updated, averaged over all iterations except the first one. We exclude the first iteration from the computation, because, in this iteration, the lazy edge selection computes the ratios of all edges (see Section 7), which implies that the savings are always zero. To compute *Average savings* for IGDE$_{\text{LAZY}}$, we perform the same computation (i.e., calculate the percentage of edges whose ratio is not updated, averaged over all iterations except the first one) on the activation graph $\tilde{G}_v$ of each vulnerable node v, and then compute the average of all results.

We first report the runtime. Fig. 9a shows that AGDE$_{\text{LAZY}}$ and IGDE$_{\text{LAZY}}$ were faster than both AGDE and IGDE, for all tested $|S|$ values. Specifically, AGDE$_{\text{LAZY}}$ was at least 37% and up to 600% faster than AGDE, while IGDE$_{\text{LAZY}}$ was at least 49% and up to 87% faster than IGDE. In addi-

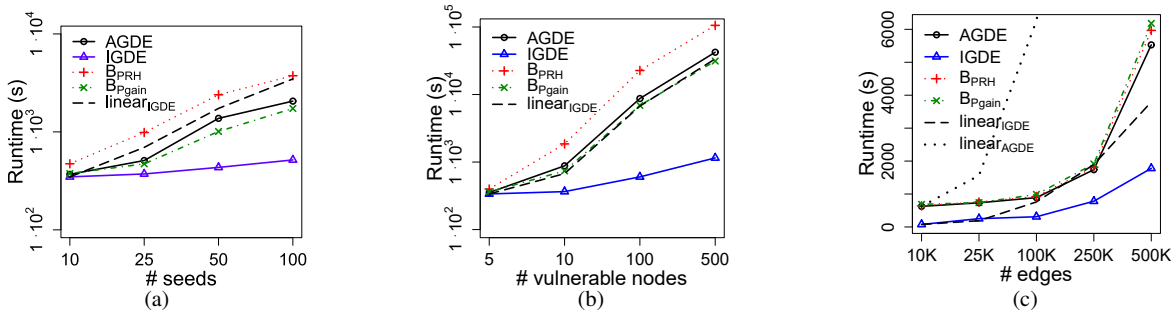


Fig. 8: (a) Runtime vs. number of seeds $|S|$, for the *Wiki* dataset. (b) Runtime vs. number of vulnerable nodes $|D|$, for the *Wiki* dataset. (c) Runtime vs. number of edges $|E|$ in the graph, for the *AB* dataset.

tion, $\text{AGDE}_{\text{LAZY}}$ and $\text{IGDE}_{\text{LAZY}}$ scaled better with $|S|$ than AGDE and IGDE.

Figs. 9b and 9c show that lazy edge selection improves the runtime of both $\text{AGDE}_{\text{LAZY}}$ and $\text{IGDE}_{\text{LAZY}}$, for all tested $|D|$ values. For example, when the *Wiki* dataset was used (Fig. 9b), $\text{AGDE}_{\text{LAZY}}$ was at least 146% and up to 814% faster than AGDE, while $\text{IGDE}_{\text{LAZY}}$ was at least 15% and up to 40% faster than IGDE. In addition, $\text{AGDE}_{\text{LAZY}}$ and $\text{IGDE}_{\text{LAZY}}$ scaled better with $|D|$ than AGDE and IGDE.

These results show that the lazy edge selection strategy improves the runtime and scalability of both AGDE and IGDE. This is because it avoids computing the ratio of a large percentage of edges after edge deletion, which is an expensive operation, particularly for large activation graphs.

To demonstrate the percentage of such edges, we report the *Average savings* measure in Fig. 10. Figs. 10a, 10b, and 10c correspond to the experiments of Figs. 9a, 9b, and 9c, respectively. As can be seen in Figs. 10a, 10b, and 10c, the *Average savings* for $\text{AGDE}_{\text{LAZY}}$ were at least 96%. Thus, the lazy edge selection strategy updated the ratio of a small percentage of edges (i.e., one of the topmost edges in the priority queue remained on the top of the queue after the update), which explains the much better efficiency of $\text{AGDE}_{\text{LAZY}}$ compared to AGDE. The *Average savings* for $\text{IGDE}_{\text{LAZY}}$ were at least 57% and up to 95%. However, they were lower than those of $\text{AGDE}_{\text{LAZY}}$. This is because in some iterations IGDE was applied to very small activation graphs of fewer than 5 edges, for which lazy edge deletion had to update most of their edges, resulting is small savings.

Threshold h. We demonstrate the impact of h on the L_1 distance and on the runtime of AGDE and IGDE. Figs. 11a and 11b show that the L_1 distance decreased by 0.07% on average, for $h \leq 10^{-3}$ and substantially for larger h values. The runtime of both methods decreased significantly as h increases. Thus, setting h to 10^{-3}, as suggested in [17], allows estimating the activation probabilities accurately and efficiently. The results for $\text{AGDE}_{\text{LAZY}}$ and $\text{IGDE}_{\text{LAZY}}$ were qualitatively similar to those of AGDE and IGDE (omitted).

Benefit of using PRH vs. the L_1 distance. We demonstrate the effectiveness and efficiency of using PRH as a heuristic to capture the change to the PageRank scores, caused by edge deletion. We compared our algorithms against $\text{B}_{L1/PGain}$, a baseline that implements the greedy approach based on the L_1 distance (see Section 3).

Figs. 12a, 12b, and 12c show the results with respect to PRH, L_1 distance, and $K\tau_b$, for varying $max\mathcal{P}$, respectively. $\text{B}_{L1/PGain}$ produced the same solution when applied using $max\mathcal{P} \leq 0.1$. Furthermore, the solution was worse than the solutions of AGDE and IGDE, with respect to all tested measures. Specifically, the PRH for $\text{B}_{L1/PGain}$ was 2.7 times larger on average than that of AGDE and IGDE, and the results in terms of L_1 distance and $K\tau_b$ were qualitatively similar. In addition, $\text{B}_{L1/PGain}$ deleted on average 1.5% and 1.6% more edges than AGDE and IGDE, respectively (see Fig. 12d).

Figs. 12e and 12f show the results for varying $|S|$ and $|D|$, respectively, with respect to the L_1 distance. The L_1 distance for $\text{B}_{L1/PGain}$ was 5 times larger than that of AGDE and IGDE, on average. In addition, $\text{B}_{L1/PGain}$ was several orders of magnitude slower than our algorithms, because it computes the PageRank distribution of the graph after deleting each edge, in order to select the best edge in each iteration. For example, $\text{B}_{L1/PGain}$ required 12 hours when applied to *Pol* with $|D| = 50$, while IGDE needed 90 seconds.

Thus, PRH is a more effective and efficient measure to avoid changes to the PageRank distribution compared to the L_1 distance.

Benefit of computing PRH on G. We demonstrate that computing the PRH of every edge in a subset E' on the graph G helps efficiency and does not impact effectiveness. We compared AGDE with $\text{B}_{PRHupd/PGain}$, a baseline that

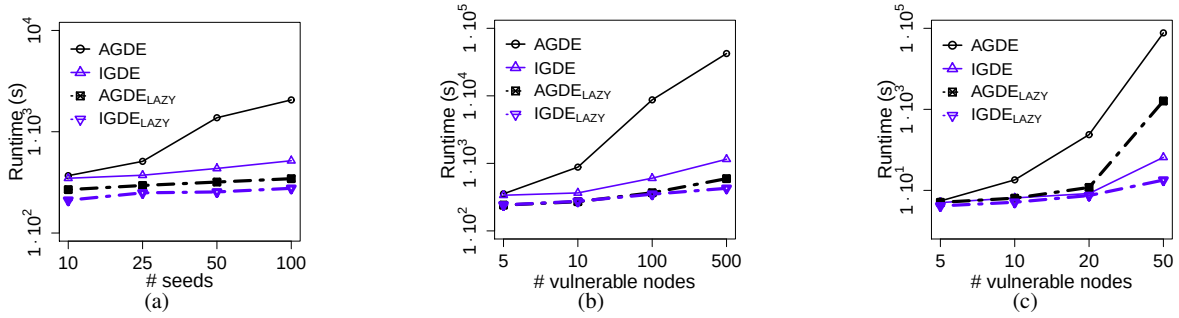


Fig. 9: Comparison of AGDE$_{\text{LAZY}}$ and IGDE$_{\text{LAZY}}$, which incorporate lazy edge selection, with AGDE and IGDE in terms of runtime. (a) Runtime vs. number of seeds $|S|$, for the *Wiki* dataset. (b) Runtime vs. number of vulnerable nodes $|D|$, for the *Wiki* dataset. (c) Runtime vs. number of vulnerable nodes $|D|$, for the *Pol* dataset.

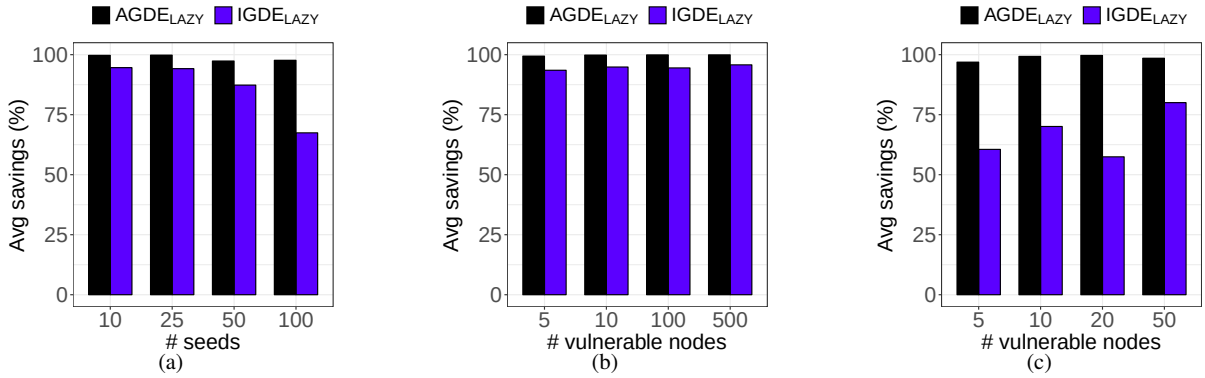


Fig. 10: Average savings (%) vs. (a) number of seeds $|S|$ for *Wiki*, (b) number of vulnerable nodes $|D|$ for *Wiki*, and (c) number of vulnerable nodes $|D|$ for *Pol*, which correspond to the experiments in Figs. 9a, 9b, and 9c, respectively. For AGDE$_{\text{LAZY}}$, *Average savings* is computed as the percentage of edges of the activation graph $\tilde{G}_D$ whose ratio is not updated, averaged over all iterations except the first one. For IGDE$_{\text{LAZY}}$, *Average savings* is computed by additionally averaging over the vulnerable nodes.

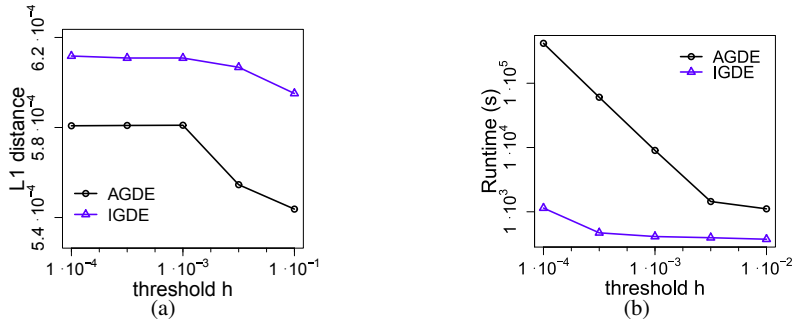


Fig. 11: (a) L_1 distance vs. threshold h, for the *Wiki* dataset. (b) Runtime vs. threshold h, for the *Wiki* dataset.

computes the PRH of an edge e on a graph G'_e, produced from G by deleting all edges that were added into E' before e. The baseline is based on AGDE, because AGDE computes the PRH of more edge subsets (potential solutions) than IGDE, and this allows comparing the PRH computation strategies on more subsets. We repeated all experiments of the effectiveness subsection above and found that AGDE and B$_{PRHupd/PGain}$ produced the same solutions, except in the experiments of Figs. 5c and 6d.

Tables 2 and 3 correspond to the experiments of Fig. 5c and 6d, respectively, and show the PRH as well as the *precision* and *recall* of the solutions of AGDE and of the B$_{PRHupd/PGain}$ baseline. The *precision* is computed as $\frac{|E'_{\text{AGDE}} \cap E'_B|}{|E'_{\text{AGDE}}|}$, and the *recall* is computed as $\frac{|E'_{\text{AGDE}} \cap E'_B|}{|E'_B|}$, where

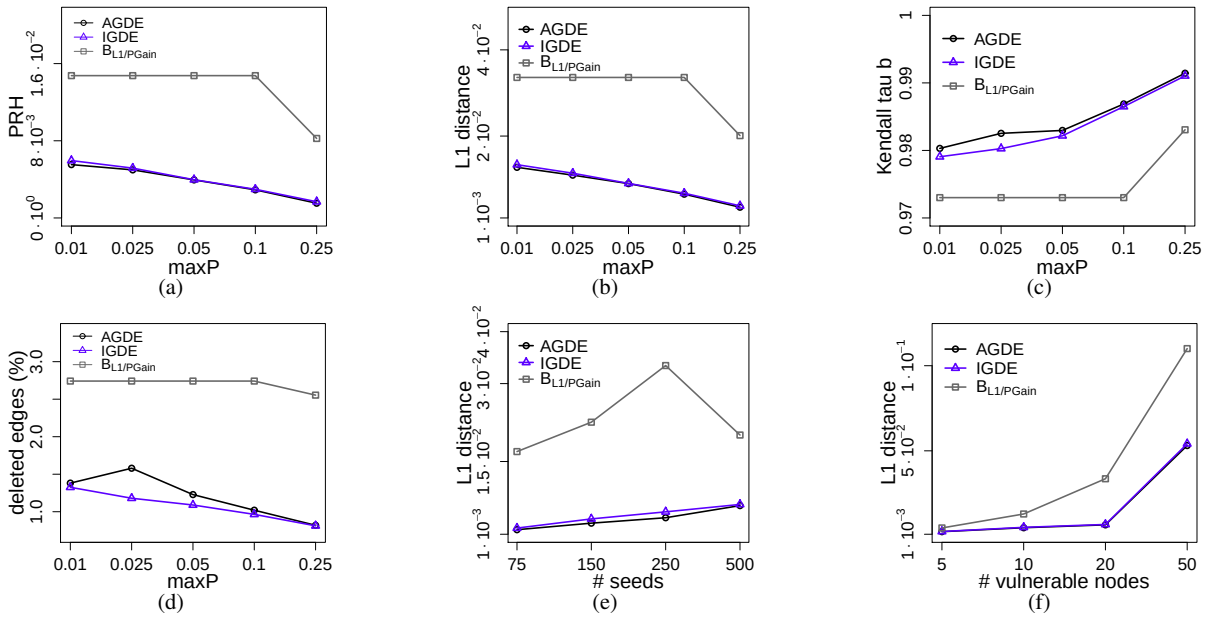


Fig. 12: Comparison with the $\mathrm{B}_{L1/PGain}$ baseline, which implements the greedy approach based on the L_1 distance (see Section 3), on the *Pol* dataset. (a) PRH vs. threshold $max\mathcal{P}$. (b) L_1 distance vs. threshold $max\mathcal{P}$. (c) Kendall τ_b correlation ($K\tau_b$) vs. threshold $max\mathcal{P}$. (d) Percentage of deleted edges vs. threshold $max\mathcal{P}$. (e) L_1 distance vs. number of seeds $|S|$. (f) L_1 distance vs. number of vulnerable nodes $|D|$.

E'_{AGDE} is the solution of AGDE and E'_B is the solution of $\mathrm{B}_{PRHupd/PGain}$.

$max\mathcal{P}$	Precision	Recall	$PRH(E'_{\mathrm{AGDE}})$	$PRH(E'_B)$
0.01	1	1	$4.7{\cdot}10^{-3}$	$4.7{\cdot}10^{-3}$
0.025	0.996	0.996	$4.235{\cdot}10^{-3}$	$4.236{\cdot}10^{-3}$
0.05	1	1	$3.37{\cdot}10^{-3}$	$3.37{\cdot}10^{-3}$
0.1	1	1	$2.477{\cdot}10^{-3}$	$2.477{\cdot}10^{-3}$
0.25	1	1	$1.298{\cdot}10^{-3}$	$1.298{\cdot}10^{-3}$

Table 2: Comparison of AGDE with the $\mathrm{B}_{PRHupd/PGain}$ baseline, on the *Pol* dataset, for varying threshold $max\mathcal{P}$ in $[0.01, 0.25]$. $\mathrm{B}_{PRHupd/PGain}$ computes the PRH of an edge e on a graph G'_e, which is produced from G by deleting all edges that were added into E' before e.

$\lvert S\rvert$	Precision	Recall	$PRH(E'_{\mathrm{AGDE}})$	$PRH(E'_B)$
10	0.991	0.991	$7.881{\cdot}10^{-4}$	$7.882{\cdot}10^{-4}$
25	0.978	0.970	$1.292{\cdot}10^{-3}$	$1.293{\cdot}10^{-3}$
50	0.978	0.972	$1.774{\cdot}10^{-3}$	$1.775{\cdot}10^{-3}$
100	1	1	$1.298{\cdot}10^{-3}$	$1.298{\cdot}10^{-3}$

Table 3: Comparison of AGDE with the $\mathrm{B}_{PRHupd/PGain}$ baseline, on the *Pol* dataset, for varying number of seeds $|S|$ in $[10, 100]$. $\mathrm{B}_{PRHupd/PGain}$ computes the PRH of an edge e on a graph G'_e, which is produced from G by deleting all edges that were added into E' before e.

As can be seen in Tables 2 and 3, the precision and recall are both equal to 1 (i.e., AGDE and $\mathrm{B}_{PRHupd/PGain}$ produced the same solution), or close to 1. The difference between the solutions was small and occurred because the algorithms broke ties differently. Specifically, the solutions differed in at most 5 edges and had similar PRH. However, AGDE was up to 84% faster, because it avoids recomputing the PageRank distribution of the graph. Thus, our PRH computation strategy is both effective and efficient.

9 Related work

Our work is related to methods that limit the diffusion of information in order to minimize the spread (expected number of nodes that are activated by a seed set). These methods can be split into two classes. The first class contains methods that modify the graph and is reviewed in Section 9.1. The second class contains methods that initiate the diffusion of information of opposite content and is reviewed in Section 9.2. In addition, our work is related to methods that modify the graph to address optimization problems related to PageRank and robustness. These methods are discussed in Section 9.3.

9.1 Minimizing spread by graph modification

Methods that modify the graph aim to minimize the spread either directly, or indirectly by optimizing a graph property.

To minimize the spread directly, there are heuristics that apply node [45, 46] or edge [26, 28] deletion under different diffusion models, such as the Independent Cascade (IC) model (e.g., [45, 46]), the Linear Threshold (LT) model (e.g., [26]), and the discrete dynamic systems model [28]. For example, the heuristics developed in [45,46] aim to find a node subset of given size whose removal from the graph minimizes the spread. These heuristics employ the dominator tree of the graph [32] to reduce the search space and improve efficiency. There is also an approximation algorithm [25] to minimize the spread directly. The algorithm aims to delete an edge subset of given size under the LT model, and it is based on the the *supermodularity* of the spread function after edge deletion [3].

Methods for minimizing the spread indirectly were proposed in [15, 21, 38, 40, 44]. These methods delete edges [40, 44] or nodes [15, 38] aiming to minimize graph properties that control the spread, such as the leading eigenvalue of the adjacency matrix [38, 40], the node degree [44], and the node betweeness (average number of shortest paths that pass through a node or edge) [15].

All methods that modify the graph follow the *collective* approach, which requires reducing the activation probabilities of *all* nodes as much as possible. In addition, they assume that deleting each edge has the same impact on the information propagation properties of the graph. Thus, these methods are not applicable to our problem, as discussed in Introduction.

9.2 Minimizing spread by diffusing information of opposite content

Methods that minimize the spread of undesirable (negative) information, by diffusing information of opposite content (positive information) were proposed in [7, 20]. These methods select seeds which diffuse the positive information, in order to prevent the diffusion of negative information to the largest (expected) number of nodes.

Specifically, the work of [7] proposes three heuristics for selecting a subset of nodes of given size, under an extended IC model. The first heuristic selects the nodes with the largest degree as seeds, the second heuristic selects nodes that are expected to be activated early as seeds, and the third heuristic selects nodes that would activate the highest number of nodes after being activated early as seeds. On the other hand, the work of [20] proposes a heuristic for selecting a subset of nodes of given size, under an extended LT model. The crux of the heuristic is the approximate computation of spread using directed acyclic graphs around seeds, which improves efficiency.

The work of [37] considered a different setting than that of [7, 20], in which the spread of the negative information must be limited to at most a threshold within a given time period. In addition, it proposed heuristics for the IC and the LT model. The heuristics first find the community structure of the graph and then perform seed selection in each community independently. This strategy can substantially improve efficiency when the graph has relatively small communities.

The problems considered in the methods of [7,20,37] are fundamentally different from *PED*. First, these problems require selecting a set of nodes as seeds. On the other hand, *PED* assumes that the set of seeds is given, and it requires selecting a set of edges. Second, these problems do not distinguish between vulnerable and non-vulnerable nodes. On the other hand, *PED* considers a set of vulnerable nodes and seeks to limit the diffusion of information to them, while not affecting the information propagation to other nodes. Therefore, the methods proposed in [7, 20, 37] cannot be applied to deal with the *PED* problem.

9.3 Graph modification for PageRank and robustness manipulation

Our work is also related to approaches that modify the graph to address optimization problems related to PageRank. These approaches perform edge deletion [13, 22] or node aggregation [33] to improve the PageRank scores of nodes, or to speed up the computation of PageRank [18]. In addition, our work is related to approaches for degrading graph robustness (resilience to changes), in the sense that they also modify the graph to prevent the communication between nodes. Approaches for degrading graph robustness perform node or edge deletion and include heuristics [1, 23], as well as approximation algorithms [8]. However, none of the aforementioned approaches [1,8,13,18,22,23,33] considers information diffusion.

10 Conclusions

Existing works for limiting the diffusion of information by edge deletion assume that the diffusing information can affect all nodes and that deleting each edge has the same impact on the information propagation properties of the graph. In this work, we introduced an approach that lifts these restrictive assumptions. Our approach reduces the activation probabilities of vulnerable nodes to at most a specified threshold, and it applies edge deletion while preserving the information propagation properties of the graph, by avoiding

[3] This function can be written as $\sum_{v\in G}(\mathcal{P}(v, Q_{S,v}, E)) - \sum_{v\in G}(\mathcal{P}(v, Q_{S,v}, E'))$, which is supermodular. This is because: (I) $\sum_{v\in G}\mathcal{P}(v, Q_{S,v}, E)$ is constant, (II) $\sum_{v\in G}(\mathcal{P}(v, Q_{S,v}, E'))$ is submodular (as a sum of submodular functions [27]), which implies that $-\sum_{v\in G}(\mathcal{P}(v, Q_{S,v}, E'))$ is supermodular [25], and (III) the sum of a constant and a supermodular function is supermodular [27].

changes to its PageRank distribution. We proposed a measure to capture these changes, and based on the measure we formulated the *PED* problem. To deal with the problem, we developed an effective approximation algorithm and an efficient heuristic. In addition, we have proposed a lazy edge selection strategy and experimentally shown its ability to substantially speed up our algorithm and heuristic.

Conflict of interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. R. Albert, H. Jeong, and A. Barabasi. Error and attack tolerance of complex networks. *Nature*, pages 378–382, 2000.
2. K. Avrachenkov and N. Litvak. The effect of new links on google pagerank. *Stochastic Models*, 22(2):319–331, 2006.
3. K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova. Monte carlo methods in pagerank computation. *SIAM J. Numer. Anal.*, 45(2):890–904, 2007.
4. P. Berkhin. A survey on pagerank computing. *Internet Math.*, 2(1):73–120, 2005.
5. P. Boldi, M. Santini, and S. Vigna. Paradoxical effects in pagerank incremental computations. *Internet Math.*, 2(3):387–404, 2005.
6. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Com. Net. & ISDN*, 30(1):107–117, 1998.
7. C. Budak, D. Agrawal, and A. E. Abbadi. Limiting the spread of misinformation in social networks. In *WWW*, pages 665–674, 2011.
8. H. Chan, L. Akoglu, and H. Tong. Make it or break it: Manipulating robustness in large networks. In *SDM*, pages 325–333, 2014.
9. W. Chen et al. Influence maximization in social networks when negative opinions may emerge and propagate. In *SDM*, pages 379–390, 2011.
10. W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD*, pages 199–208, 2009.
11. W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *ICDM*, pages 88–97, 2010.
12. V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.
13. B. Csáji, R. Jungers, and V. Blondel. Pagerank optimization by edge selection. *Discrete Appl. Math.*, 169:73–87, 2014.
14. T. Fujito. Approximation algorithms for submodular set cover with applications. *IEICE Trans. Inf. and Syst.*, e83(3), 2000.
15. C. Gao, J. Liu, and N. Zhong. Network immunization and virus propagation in email networks: experimental evaluation and analysis. *KAIS*, 27(2):253–279, 2011.
16. A. Goyal, W. Lu, and L. V. S. Lakshmanan. CELF++: Optimizing the greedy algorithm for influence maximization in social networks. In *WWW*, pages 47–48, 2011.
17. A. Goyal, L. Wei, and L. Lakshmanan. SIMPATH: An efficient algorithm for influence maximization under the linear threshold model. In *ICDM*, pages 211–220, 2011.
18. M. Gupta, A. Pathak, and S. Chakrabarti. Fast algorithms for topk personalized pagerank queries. In *WWW*, pages 1225–1226, 2008.
19. S. Gupta. A conceptual framework that identifies antecedents and consequences of building socially responsible international brands. *Thunderbird International Business Review*, 2015.
20. X. He, G. Song, W. Chen, and Q. Jiang. Influence blocking maximization in social networks under the competitive linear threshold model. In *SDM*, pages 463–474, 2012.
21. H. W. Hethcote. The mathematics of infectious diseases. *SIAM Rev.*, 42(4):599–653, 2000.
22. I. Hideaki and T. Roberto. Computing the pagerank variation for fragile web data. *SICE Journal of Control, Measurement, and System Integration*, 2(1):1–9, 2009.
23. P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han. Attack vulnerability of complex networks. *Phys. Rev. E*, 65:056109, 2002.
24. D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.
25. E. B. Khalil, B. Dilkina, and L. Song. Scalable diffusion-aware optimization of network topology. In *KDD*, pages 1226–1235, 2014.
26. M. Kimura, K. Saito, and H. Motoda. Solving the contamination minimization problem on networks for the linear threshold model. In *PRICAI*, pages 977–984, 2008.
27. A. Krause and D. Golovin. Submodular function maximization. In *Tractability*. Cambridge University Press, 2013.
28. C. Kuhlman, G. Tuli, S. Swarup, M. V. Marathe, and S. S. Ravi. Blocking simple and complex contagion by edge removal. In *ICDM*, pages 399–408, 2013.
29. A. N. Langville and C. D. Meyer. A survey of eigenvector methods for web information retrieval. *SIAM Review*, 47(1):135–161, 2005.
30. J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.
31. H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *HLT*, pages 912–920, 2010.
32. E. S. Lowry and C. W. Medlock. Object code optimization. *Commun. ACM*, 12(1):13–22, 1969.
33. J. M. Maestre and H. Ishii. A cooperative game theory approach to the pagerank problem. In *American Control Conference*, pages 3820–3825, 2016.
34. M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *IFIP Conference on Optimization Techniques*, pages 234–243, 1978.
35. B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause. Lazier than lazy greedy. In *AAAI*, pages 1812–1818, 2015.
36. G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.
37. N. P. Nguyen, G. Yan, M. Thai, and S. Eidenbenz. Containment of misinformation spread in online social networks. In *ACM WebSci*, pages 213–222, 2012.
38. S. Saha, A. Adiga, B. A. Prakash, and A. K. S. Vullikanti. Approximation algorithms for reducing the spectral radius to control epidemic spread. In *SDM*, pages 568–576, 2015.
39. N. C. Smith and E. Cooper-Martin. Ethics and target marketing: The role of product harm and consumer vulnerability. *Journal of Marketing*, 61(3):pp. 1–20, 1997.
40. H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *CIKM*, pages 245–254, 2012.
41. L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.
42. B. Xiang et al. Pagerank with priors: An influence propagation perspective. In *IJCAI*, pages 2740–2746, 2013.
43. S. Yu, G. Gu, A. Barnawi, S. Guo, and I. Stojmenovic. Malware propagation in large-scale networks. *IEEE TKDE*, 27(1):170–179, 2015.

44. H. Zhang, K. Li, X. Fu, and B. Wang. An efficient control strategy of epidemic spreading on scale-free networks. *Chinese Physics Letters*, 26(6):068901, 2009.
45. Y. Zhang and B. A. Prakash. DAVA: distributing vaccines over networks under prior information. In *SDM*, pages 46–54, 2014.
46. Y. Zhang and B. A. Prakash. Data-aware vaccine allocation over large networks. *ACM TKDD*, 10(2):1–32, 2015.